POWER GRID VERIFICATION AND OPTIMIZATION

BY

TING YU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Professor Martin D. F. Wong, Chair
Professor Andreas Cangellaris
Associate Professor Deming Chen
Professor Elyse Rosenbaum
Professor Jose E. Schutt-Aine

ABSTRACT

IR-drop is the voltage drop that is caused by the impedance of power grid and devices' switchings. It is important to verify voltage values of nodes on power grids. To make the circuit work reliably, it is preferable to reduce the voltage drops. Nowadays, power grids are usually in large size, which results in the runtime and memory bottleneck with traditional methods. In order to address these issues, we focus on developing efficient methods to perform power grid verification and optimization.

There are three topics related to power grid verification. Based on the distributed memory system, we propose an efficient parallel domain decomposition method for power grid DC analysis. The largest power grid size that can be solved is not limited by the memory of a single processor. We develop an efficient method to balance the data load of all the processors. Only voltage values of boundary nodes are extracted and exchanged for data communication. The communication overhead is minimal. With over 1000 processors, the proposed method achieves a 110X speedup over a state-of-art LU solver. A power grid with 192 M nodes can be processed within minutes.

To accelerate the power grid transient analysis, we present PGT_SOLVER. This direct method based solver is developed on a shared memory system. Advanced techniques such as sparse vector and solution mapping are developed or utilized to accelerate the forward and backward substitutions in each time step. Multiple threads are utilized to further reduce runtime. As the first-place winner in the "TAU_2012 power grid simulation contest", PGT_SOLVER effectively reduces the runtime of transient analysis without introducing any error. Memory consumption of this solver is also very affordable.

Combining the flow of parallel DC analysis and techniques of PGT_SOLVER, we develop an effective parallel method for power grid transient analysis. Special considerations are made to achieve better performance, such as power

grid partitioning. With only a few hundred processors, over 69X speedup is achieved compared to the sequential PGT_SOLVER. To alleviate the memory usage of solving a large size power grid, the parallel process can be operated in multiple steps. With fewer processors, the propose method is still capable of performing efficient simulation of large power grids.

Besides developing parallel solvers to accelerate DC and transient analysis, we also explore a few methods to reduce the IR-drop values of the power grids. These include the optimization of power pads and the on-chip low-dropout voltage regulator (LDO). With a fixed number of power pads, we develop a method to relocate the pads to optimize DC IR-drop values. A novel IR-drop driven method is proposed to calculate effective locations of the power pads. Moving pads to these locations effectively reduces the IR-drop values. Multiple power pads are moved simultaneously, which accelerates the optimization. Within limited iterations, IR-drop values of the power grids are effectively reduced.

By integrating the on-chip low-dropout voltage regulator, transient IR-drop values can be reduced. We propose a simulation-based method to integrate LDOs into the power grids. A hybrid flow is utilized to perform the transient analysis. The Cholesky direct solver and SPICE are utilized in the simulation flow. With an effective optimization method, a set of LDOs is added into the power grids and placed at locations which effectively reduce transient IR-drop values.

To my father, Liuqing Yu
my mother, Meizhi Zheng
my husband, Yubo Yan
for their endless love and support

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Professor Martin D. F. Wong, for his continuous support of my Ph.D study and research. His patience, motivation, enthusiasm, and immense knowledge are not only beneficial for my research, but also will favorably contribute to all my life. It is an honor to be one of his students.

My sincere thanks also go to all of my colleagues: Hongbo Zhang, Qiang Ma, Yuelin Du, Peici Wu, Leslie Huang, Zigang Xiao, Haitong Tian, Daifeng Guo and Choden Konigsmark. It is a pleasure to know and work with them. Those discussions and the time spend together make the life here full of memories. Their encouragement was invaluable for my Ph.D. study.

Last but not the least, I would like to thank my parents Liuqing Yu and Meizhi Zheng for loving me throughout my life. I also thank my husband Yubo Yan for always providing strong support to me.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
1.1 Power grid analysis 2
1.2 Power grid optimization 5

CHAPTER 2 EFFICIENT PARALLEL POWER GRID DC ANALYSIS VIA ADDITIVE SCHWARZ METHOD 8
2.1 Geometrical additive Schwarz method (ASM) 9
2.2 Parallel ASM on distributed memory system 10
2.3 Experimental results 17
2.4 Summary 19

CHAPTER 3 PGT_SOLVER: AN EFFICIENT SOLVER FOR POWER GRID TRANSIENT ANALYSIS 21
3.1 Equivalent companion models of capacitance and inductance . 22
3.2 Methodology 24
3.3 Experimental results 30
3.4 Summary 33

CHAPTER 4 EFFICIENT PARALLEL SOLVER FOR POWER GRID TRANSIENT ANALYSIS 34
4.1 Flow of the proposed method 35
4.2 Partitioning of the power grid 36
4.3 Experimental results 40
4.4 Summary 46

CHAPTER 5 A NOVEL AND EFFICIENT METHOD FOR POWER PAD PLACEMENT OPTIMIZATION 47
5.1 Method 47
5.2 Experimental results 55
5.3 Summary 60

CHAPTER 6 EFFICIENT SIMULATION-BASED OPTIMIZATION OF POWER GRID WITH ON-CHIP VOLTAGE REGULATOR 61
6.1 LDO's feature and supply voltage of power grid 62

6.2 Hybrid simulation of power grid with LDOs 63
6.3 Optimizing power grid with LDOs 64
6.4 Experimental results 69
6.5 Future works 73
6.6 Summary 73

CHAPTER 7 CONCLUSIONS 74

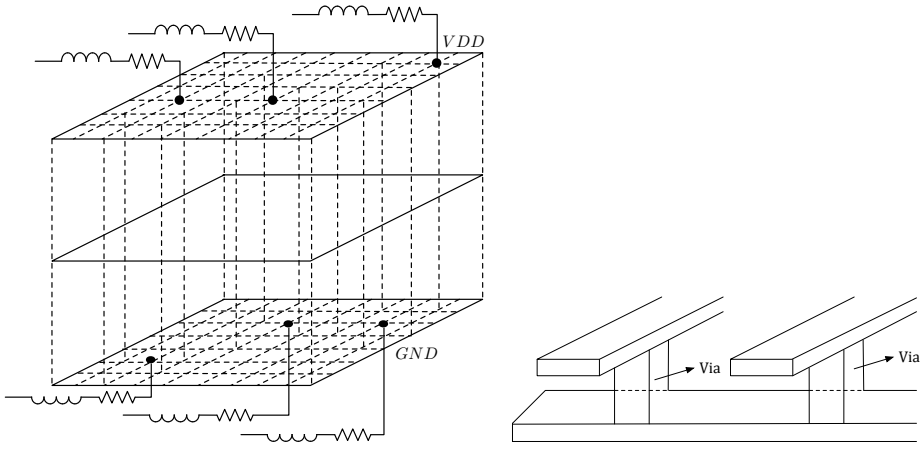
REFERENCES 76

CHAPTER 1

INTRODUCTION

A power grid is an important component of the power delivery system. It provides power to on-chip devices through metal wires and vias. An example power grid is illustrated in Figure 1.1(a).

The middle layer of Figure 1.1(a) represents the device layer. The top and bottom layers represent the power and ground networks. With multi-voltage techniques, there are some extra networks providing the chip with multiple voltage values. Both the top and bottom layers can be modeled as resistor networks or RLC networks. Here they are simplified into two-dimensional grids. They can also be modeled as three-dimensional grids, which is more accurate. The three-dimensional grid is composed of multiple layers of metal wires. Each layer of metal wires goes along a horizontal or vertical direction. Different layers of metal wires are connected by vias, which are shown in Figure 1.1(b).



(a) an example power grid

(b) connection of metal wires of different layers

Figure 1.1: An example power grid

Due to the impedance of the power grid, there is a voltage drop across the

grid, which is commonly referred to as IR-drop or voltage drop. IR-drop has a great impact on circuit performance. Excessive voltage drops in the power grid reduce switching speeds and noise margins of circuits. They may even lead to functional failures of circuits. As a result, it is important to verify the voltage values of nodes of the grid and optimize the power grid to reduce the IR-drop values.

Because of the fast development of technology, the power grid is becoming larger and larger. A power grid usually contains tens of millions of nodes. It is even possible for a power grid to contain hundreds of millions of nodes. The large size of the power grid brings big challenges to both power grid analysis and optimization.

1.1 Power grid analysis

There are two types of power grid analysis: DC analysis and transient analysis. DC analysis is utilized to obtain the static IR-drop values, which are caused by the resistance of interconnects. Transient analysis is utilized to obtain dynamic IR-drop values, which are caused by inductance and the switching of the devices.

1.1.1 Parallel DC analysis based on distributed system

In DC analysis, the power grid is modeled as a resistor network. Performing DC analysis is equivalent of solving a linear system. The large size of the power grid causes long runtime and memory bottleneck for traditional sequential power grid solvers. With more advanced methods, such as random walks and PCG [1, 2], the above challenges still exist. For example, it still costs a single processor 8 hours and 107 GB memory to simulate a power grid with 136 million nodes [3].

There are quite a few parallel works developed for power grid analysis, including the GPU-based multigrid method [4], the supernodal method based on a shared memory system [5, 6] and the domain decomposition method based on distributed system [7,8]. However, the disadvantages of these methods limit their efficiency. For example, the multigrid method may introduce error in final results. The efficiency of the methods developed on a shared

memory system is restricted by the limited number of threads. Besides, both the GPU and shared memory system have a limited memory size, which cannot eliminate the memory bottleneck caused by the large size of the power grid.

The distribute memory system can overcome the memory challenge by including more processors into the system. However, the parallel domain decomposition methods such as those presented in [7,8] are not efficient in data storage and communication. In the above methods, a master processor stores the whole system conductance matrix and then distributes a set of submatrices into other processors. The data storage load between the master processor and other processors is very uneven. Besides, the data communications of these methods are not efficient. Instead of only exchanging the voltage values of boundary nodes, voltage values of all the nodes of each subdomain are exchanged, which introduces much higher communication overhead.

In Chapter 2, we propose an efficient parallel method to perform DC analysis of a large size power grid. It is an iterative method that is developed on the distributed memory system. It is based on the geometrical domain decomposition method. We divide the power grid into a set of blocks or subdomains. Overlapping between blocks is introduced to accelerate the convergence. We develop a new data storage method that successfully overcomes the memory bottleneck. We also propose an efficient method to extract and exchange the voltage values of boundary nodes of each subdomain. Minimum communication overhead is achieved.

1.1.2 Accelerating transient analysis with shared memory system

In transient analysis, the power grid is modeled as a RLC network. Performing power grid transient analysis is equivalent of solving a linear systems at each time step. Transient analysis is more accurate than DC analysis, but is also more time consuming. As a result, it is more urgent to reduce the runtime of transient analysis.

Although the previous parallel method brings a lot of speedups for DC analysis, it may be less effective to reduce the runtime of transient analysis. This is due to the difference in runtime bottlenecks of DC and transient

analysis. For DC analysis, matrix decomposition is most time consuming. An effective method to reduce the runtime of DC analysis is to divide the matrix into a set of submatrices. Solutions are then obtained through iterative methods [4,6–10]. For transient analysis, however, it is possible that the forward and backward substitutions cost more time than matrix decomposition. The previous iterative method cannot be directly utilized to reduce the runtime of transient analysis. Actually, it can even make the transient analysis slower, because it greatly increases the cost of forward and backward substitutions. The runtime overhead accumulates along the time steps.

In Chapter 3, we present a direct method based solver to reduce the runtime of transient analysis. It is called PGT_SOLVER. It is developed on a shared memory system. The largest power grid size that can be solved depends on the memory size of a single processor. We only build and factorize the global conductance matrix limited times, and we utilize the factorized matrix for all the time steps. We propose several techniques to speedup the simulation, including the sparse vector technique, the solution mapping technique and the memorized supernode technique. An effortless but effective parallel method with multiple threads is introduced to further reduce the runtime.

1.1.3 Parallel transient analysis based on distributed memory system

Although advanced techniques are proposed in PGT_SOLVER, only limited speedup is achieved over direct solvers. Considering that transient analysis is time-consuming, it is necessary to develop an efficient parallel method to reduce the runtime of transient analysis. In Chapter 4, based on the distributed memory system, we combine parallel DC flow and techniques of PGT_SOLVER and develop an efficient parallel solver for power grid transient analysis. Special considerations such as power grid partitioning are made to achieve better performance of transient analysis. The method is applicable for users with several hundred processors. To reduce the computational memory usage in solving a large size power grid, the parallel operations can be executed in multiple steps. With this technique, fewer processors are required to perform the analysis.

1.2 Power grid optimization

In order to make the circuit work reliably, IR-drop values of the power grid should be less than some value. The constraint is shown in equation (1.1). If the IR-drop values do not meet the constraint, it is necessary to perform some optimization to reduce the IR-drop values.

$$\Delta V_{max} \leq 10\% VDD \tag{1.1}$$

In equation (1.1), ΔV_{max} is the maximum IR-drop value of nodes of the power grid. VDD is the supply voltage.

1.2.1 Power pad placement optimization

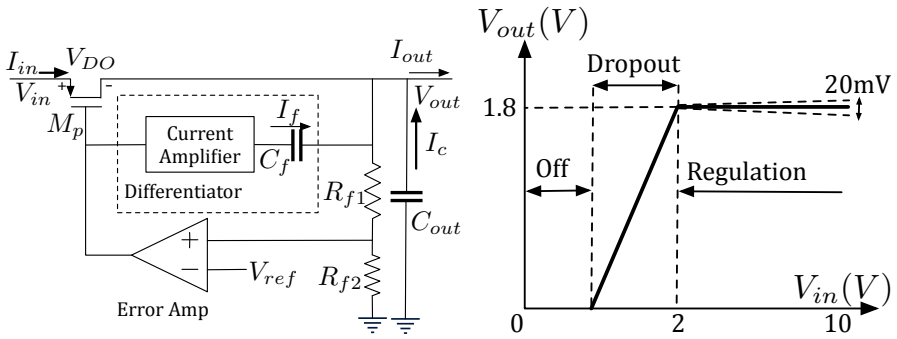
Traditional methods for reducing IR-drop values include widening metal wires to reduce the resistance of the interconnection and inserting decoupling capacitance to reduce the instant current fluctuation [11]. Another method is optimizing the numbers and locations of power supply pads. By directly controlling the locations of power pads, this method is more effective in reducing the IR-drop values.

Due to the increased size of the power grid, it is challenging to explore the effective locations of power pads to reduce the IR-drop values of the grid, especially with flip chip packages. Zhong and Wong [12] proposed a Simulated Annealing (SA) method to optimize the placement of $C4$ bump arrays. Errors may be introduced and it is not efficient in performing the power pad placement optimization.

In Chapter 5, with fixed number of power pads, we propose an efficient iterative method for power pad placement optimization. The object is to reduce static IR-drop values. We develop a novel IR-drop driven method to calculate the new locations of all the pads. Moving pads to these new locations reduces local IR-drop values. We develop a graph-based method to move multiple pads to their new locations, which reduces the global IR-drop values. After relocating the pads, IR-drop values are updated with a direct method.

1.2.2 Optimizing the number and locations of on-chip LDO

In recent years, progress in the development of on-chip low-dropout voltage regulator (LDO) has increased. An example LDO and its characteristic curve are shown in Figure 1.2(a) and Figure 1.2(b). Studies have shown that this on-chip device improves load regulation, reduces crosstalk, eliminates load-transient spikes and saves board space [13]. Integrating this device into the power grid has many benefits. For example, Zeng et al. [14] demonstrated the significant benefits of on-chip LDOs in suppressing both high-frequency switching noises and mid-frequency drops caused by resonance. The effect of the LDOs in reducing the noises is shown in Figure 1.3. Besides, authors of [15] explored the stability analysis of integrating LDOs into power delivery system.



(a) LDO structure (b) $V_{in} - V_{out}$ curve of LDO

Figure 1.2: Structure and electrical $V_{in} - V_{out}$ curve of a LDO

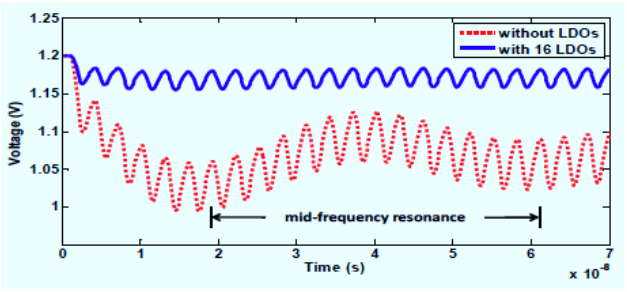


Figure 1.3: Effect of LDOs in reducing power delivery system noises

Considering that the distributions of the on-chip low-dropout voltage regulators directly affect the voltage values of nodes in the power grid, it is necessary to explore the number and locations of the on-chip LDOs. There are several challenges of optimizing the number and locations of LDOs to

meet the IR-drop constraint. First, due to the nonlinear and active feature, there is no public numerical computation model available for LDO. Traditional power grid solvers cannot simulate the system of the power grid and LDOs. Besides, because performing transient analysis is time-consuming, an effective flow is required to reduce the runtime of optimization. The added LDOs should be inserted at the locations which can effectively reduce the IR-drop values of the power grid.

In Chapter 6, we explore the optimization of the low-dropout voltage regulator to meet the IR-drop constraint, where the maximum IR-drop value is less than 10% of the power supply. With the Cholesky direct solver and SPICE, we propose a method to simulate the system of the power grid with LDOs. Based on the simulation method, we develop an efficient flow to optimize the number and locations of the on-chip voltage regulators.

CHAPTER 2

EFFICIENT PARALLEL POWER GRID DC ANALYSIS VIA ADDITIVE SCHWARZ METHOD

As mentioned before, power grid DC analysis is challenging in both runtime and memory. It is necessary to develop an efficient parallel method to reduce runtime and eliminate the memory bottleneck.

In DC analysis, the power grid is modeled as resistor networks. Figure 2.1 shows the power network of the grid. Power supply pads are distributed on the top layer of the grid. They are either distributed along the power ring with wire-bond packages, or distributed at all possible positions of the network with flip chip packages. Devices are modeled as static current sources. The dotted lines in Figure 2.1 represent the metal wires. Wires on each layer go along horizontal or vertical directions. Wires on different layers are connected by vias, which are modeled as resistors in Figure 2.1.

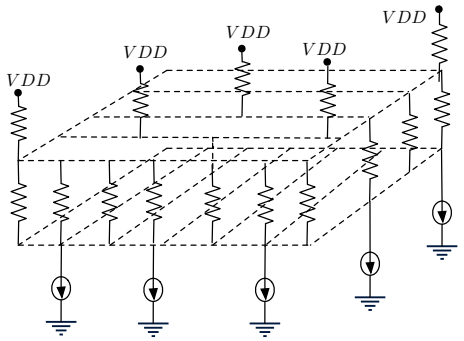


Figure 2.1: Power grid model for DC analysis

DC analysis is equivalent of solving a linear system:

$$G \times V = I \tag{2.1}$$

where G is the system conductance matrix, and V and I are the voltage and current vectors of the nodes of the power grid.

We propose an efficient parallel method for large-scale power grid analysis.

It is based on the geometrical additive schwarz method and is developed on the distributed memory system. We divide the power grid into a set of blocks or subdomains. Overlapping between subdomains is introduced to accelerate the convergence. We develop a new data storage method that successfully overcomes the memory bottleneck. A single processor is utilized to extract the net connections of each subdomain and write them into a corresponding file. All processors parse data from the files and solve the subdomains simultaneously. We also propose an efficient method to extract and exchange only the voltage values of boundary nodes of each subdomain. Minimum communication overhead is achieved.

We test the proposed method on a set of industrial benchmarks, which show that the proposed method achieves $110X$ speedup over a state-of-art LU solver. Besides, as far as we know, it is the first time reported that a power grid containing 192 M nodes can be solved within five minutes.

In the rest of this chapter, Section 2.1 and Section 2.2 present the proposed method. Experimental results and summary are listed in Section 2.3 and Section 2.4.

2.1 Geometrical additive Schwarz method (ASM)

Geometrical Additive Schwarz Method (ASM) is a type of domain decomposition method. It solves a boundary value problem for a partial differential equation approximately by splitting it into boundary value problems on smaller domains and adding the results.

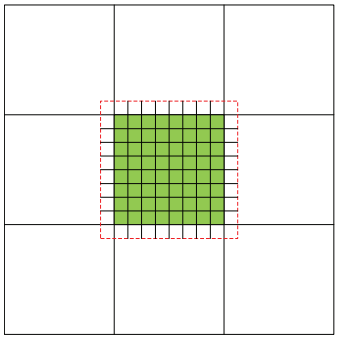


Figure 2.2: Geometrical additive Schwarz method

Figure 2.2 illustrates the geometrical ASM of a two-dimensional (2D) do-

main. The single domain is divided into a set of subdomains. Each subdomain forms a local matrix, which can be solved by both direct and iterative methods. As shown in Figure 2.2, overlapping among subdomains can be introduced. Overlapping reduces the eigenspectrum of system conductance matrix, thus leads to faster convergence [16, 17]. More overlapping leads to faster convergence. However, because of overlapping, the equivalent subdomain size is increased, which deteriorates the performance of the whole program. For optimal performance consideration, exploring the balance between the overlapping ratio and subdomain size is required.

2.2 Parallel ASM on distributed memory system

The flow of our method is shown in Figure 2.3. It is based on distributed memory system and message-passing library (MPI). The power grid is geometrically divided into a set of overlapped subdomains. A single processor extracts the information of each subdomain and writes it into an independent file. Then, all the processors parse data from the files simultaneously. In each iteration, processors first solve the subdomains in parallel, and then update the boundary information through exchanging the voltage values of the boundary nodes of each subdomain. New iteration starts if the difference between the old and new values of the nodes is bigger than some threshold value.

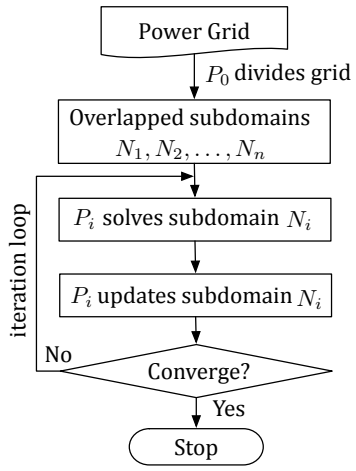


Figure 2.3: Flow chart for parallel ASM

Two key factors affect the performance of the proposed parallel ASM

method. First is the memory load for each of the processors. In order to solve a large size problem, we have to balance the data load of all the processors. Second is the data communication strategy. It affects the efficiency of the algorithm. In the following sections, we give detailed illustrations of our method with these issues considered.

2.2.1 Power grid partitioning

It is well known that for a structure with fixed dimension, the higher dimensional partitioning, the more favorable the surface-to-volume ratio and better the scalability [18, 19]. Considering that the power grid we are working on is three-dimension, it is intuitive to perform 3D partitioning of the power grid. However, this may bring convergence difficulty of the proposed flow. The reason is illustrated as follows.

The power grid structure along the top (Z) direction is different from that of the horizontal (X) and vertical (Y) directions. As mentioned before, the power grid is connected by vias along the top (Z) direction. The vias have much less resistance than that of the horizontal and vertical metal wire segments. Because of vias, the linear system can be ill-conditioned. If we also divide the grid along the Z direction, vias are exposed to the iterative process of subdomains. Zhong and Wong [20] pointed out that the iterative method may not be effective in handling the ill-conditioned system and may cost a lot of iterations to reach convergence, or even can diverge. On the other hand, the direct method is more effective in handling the ill-conditioned system [20]. As a result, we only partition the grid along the horizontal (X) and vertical (Y) directions. By doing this, vias are not exposed to the interface of different subdomains and the risk of divergence because of the partitioning is eliminated.

2.2.2 Data storage

Considering overlapping, we propose a new method to extract and store the net connections of all the subdomains.

According to the number of subdomains, overlapping ratio and the geometrical information of the power grid, we first extract geometrical boundaries

of each subdomain. Then, by extracting all the net connections of each subdomain, the master processor divides the original input file containing all the information of the whole grid into a set of different files. Each file maintains all the net connections of a subdomain. After this, all processors parse the files and solve the subdomains simultaneously. Different from the previous work [8], the master processor never stores the information of the whole grid. It only stores the subdomain it works on. As a result, the severe uneven memory load between the master and other processors is eliminated.

In the above process, special attention needs to be paid when extracting the net connections of the subdomains. Because of overlapping, a single net can belong to multiple subdomains, which is shown in Figure 2.4.

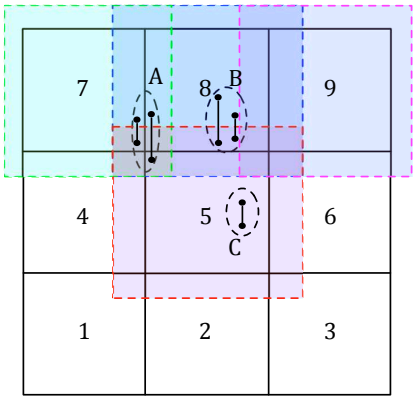


Figure 2.4: Three types of nets

Figure 2.4 shows some nets along vertical directions. There are nine blocks in this figure. Plain blocks represent subdomains without overlapping, while shadowed ones are the enlarged subdomains with overlapping.

Due to overlapping, a net can belong to four subdomains, two subdomains or a single subdomain, which are shown as the net sets A, B and C. When we are extracting these nets, we have to write them into the corresponding four files, two files or a single file.

After writing the nets into files, all the processors parse data from its corresponding file and solve the subdomains simultaneously. Both direct and iterative methods can be utilized to solve the subdomains. Here we utilize the LU direct method to solve the subdomains.

2.2.3 Data communication

After solving the subdomains and before starting a new iteration, each processor needs to update its voltage values of boundary nodes by communicating with other processors. Considering efficiency, it is only necessary to extract and exchange the voltage values of boundary nodes. However, due to overlapping, it is very challenging.

Because of overlapping, each subdomain has eight neighboring subdomains instead of four subdomains. Overlapping also shifts the locations of boundary nodes, which is shown in Figure 2.5.

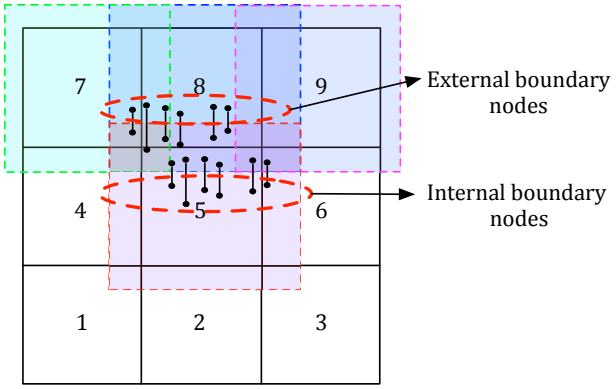


Figure 2.5: Distribution of boundary nodes

In Figure 2.5, we only show the distribution of nets along the vertical direction. The definition of blocks is the same as for Figure 2.4.

Because of overlapping, for each subdomain, there are two types of boundary nodes. The first type of boundary nodes are the nodes that do not belong to a subdomain, but are connected to internal nodes of this subdomain. The second type of boundary nodes are nodes that belong to a subdomain and are boundary nodes of neighboring subdomains. We define the first type of nodes as **external boundary nodes**, and the second ones as **internal boundary nodes**.

There are several difficulties to extract only the voltage values of boundary nodes. First, as shown in Figure 2.5, net connections between subdomains can be irregular. We also need to consider the multilayer structure of the power grid, which leads to multilayer boundary nets between subdomains. Besides, because of overlapping, several subdomains may have a copy of the boundary nodes' voltage values. After the processors solve the subdomains,

there are multiple copies of the values of these nodes. For convergence consideration, only one value should be used to update the subdomains.

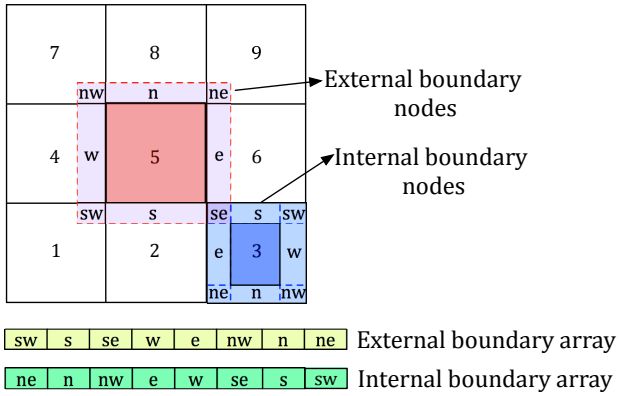


Figure 2.6: Geometrical relationship of boundary nodes in neighboring subdomains

We notice that the geometrical information of the subdomains can be utilized to overcome the above difficulties. For each processor, we allocate a local external and internal boundary array to store the voltage values of the boundary nodes. Based on the geometrical features of the boundary nodes, we classified each type of boundary nodes into eight node sets, which are shown in Figure 2.6. Each node set indicates the geometrical relationship between two neighboring subdomains. For example, in Figure 2.6, "se" is short for "south east". This node set of the external boundary of subdomain 5 stores the voltage values of a internal boundary node set of subdomain 3, which is defined as "se" node set. This definition matches the geometrical relationship between subdomain 3 and 5. Other node sets are defined in the same way. It takes O(1) time to locate the boundary node set from the geometrical relationship of neighboring subdomains.

Each segment of the local internal and external boundary arrays represents voltage values of one node set. Nodes' values within that set are in ascending order of the index of neighboring subdomains. Boundary nodes of each node set are sorted according to their three-dimensional coordinates. It is thus insensitive to the irregular and multiple layer structure of the power grid. With the geometrical relationship and coordinates, it is easy to access a boundary node's value from the boundary arrays. Time to locate a boundary node's value is $O(n_i)$, where n_i is the size of a node set.

With the above definitions, processors exchange the voltage values of

boundary nodes. When each processor has new solutions for all internal nodes of its subdomain, we first copy the internal boundary nodes' voltage values to the local internal boundary array. Processor "0" gathers these arrays from subdomains into a global internal boundary array. The global internal boundary array is reordered to generate the global external boundary array. This array contains all the external boundary nodes' voltage values for each subdomain. Processor "0" then distributes the external boundary nodes' values to all the processors, so that they obtain the updated boundary information of its subdomain. The gathering and distributing processes are done in parallel.

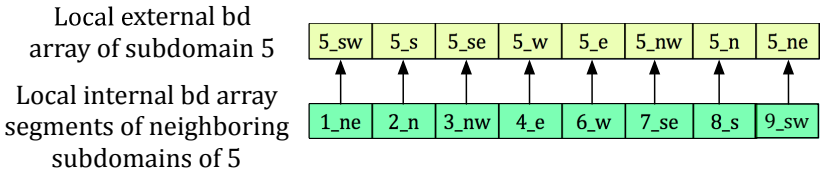


Figure 2.7: Reordering internal boundary array into external boundary array

Algorithm 1: Reordering

1 **Input:** Global internal boundary array of processor "0".
2 **Output:** Global external boundary array of processor "0".

```
// scan all subdomains
for i = 0 to num_subdomains do
  // fill in each subset of external boundary array
  for each subset in external boundary array do
    copy corresponding subset from internal boundary arrays of
    neighboring subdomains
  end for
end for
```

The reordering algorithm is shown as Algorithm 1. A detailed explanation of the internal loop is shown in Figure 2.7, where subdomain 5 is utilized as an example. The labels represent "$< subdomain\_id > \_ < node\_set >$". According to the geometrical relationships, corresponding node sets from the internal boundary array of neighboring subdomains are located and copied into the external boundary array of subdomain 5.

As mentioned before, there are multiple copies of voltage values of boundary nodes in the overlapping area. We utilize the copy from the biggest subdomain for the voltage values of these boundary nodes.

The above data exchange process has minimum communication overhead. With the message passing model (MPI), the equation for communication cost is as follows:

$$T_{msg} = t_s + t_w L \tag{2.2}$$

where t_s is the start-up time, t_w is the incremental transfer time per word and L is the length of message in words.

Generally, t_s is tens of thousands larger than t_w. Since the total number of boundary nodes is small, time spent on $t_w L$ is ignorable. As a result, it is highly preferable to reduce the number of sending and receiving operations, so that we can decrease the start-up time.

Since there are sending and receiving operations, at least two start-up times are required. With our method, only one gathering and one scattering operation are needed in each iteration. The start-up time is only $2t_s$, which is the minimum communication cost.

2.2.4 Grouping processors

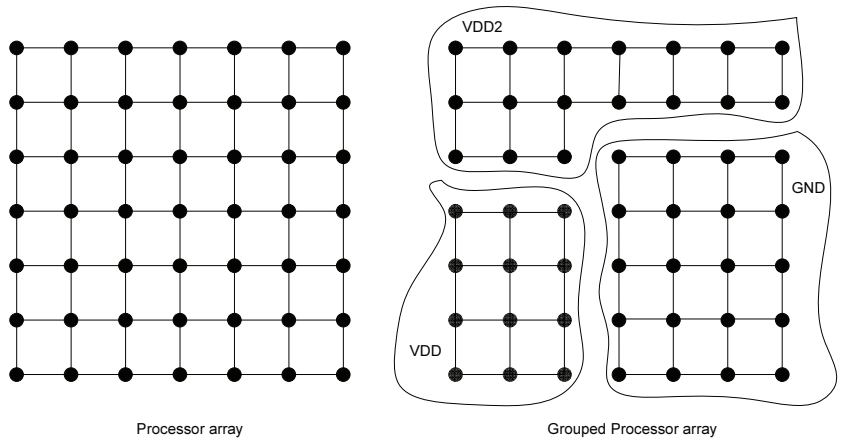


Figure 2.8: Grouping processors

As shown in Figure 1.1(a), there are VDD and GND networks in the power grid. With the multi-voltage technique, more networks such as $VDD2$ can be included in the grid. These networks are usually assumed to be disjoint

in power grid analysis and can be solved in parallel.

If we have sufficient processors, we can group the processors into several clusters. Each cluster works on one network, as shown in Figure 2.8. By this technique, we can further speed up the program. If the networks are similar in size, assuming there are three networks in the power grid, we can reduce the runtime by $\frac{2}{3}$.

2.3 Experimental results

We implement the proposed method on the distributed memory system. To make a comparison, we utilize the UMFPACK package, which is one of the best direct LU solvers on a single CPU. The program is developed in C++ under the Linux system and compiled with "mpicxx".

The parallel platform consists of approximately 300 cluster compute nodes, each with two 2.67 GHz Intel Xeon hex-core processors and 24 GB of RAM, for a total of about 3600 processors. Each processor has about 1.2 GB memory. The message passing package MPICH2 is used.

We estimate the error of each iteration as in equation (2.3), where k is the iteration number, and N is the total number of nodes in the power grid. If $residue$ is less than the stopping criteria, the program is converged.

$$residue = \max_{i=1,\ldots,N} abs(V_i^{k+1} - V_i^k) \tag{2.3}$$

The stopping criteria is set with an accuracy requirement, which is listed in Equation (2.4). V_{ref_i} is the accurate solution of each node from the direct LU solver. In order to meet the accuracy requirement for all benchmarks that LU can solve, the stopping criteria is set to be 10^{-5}.

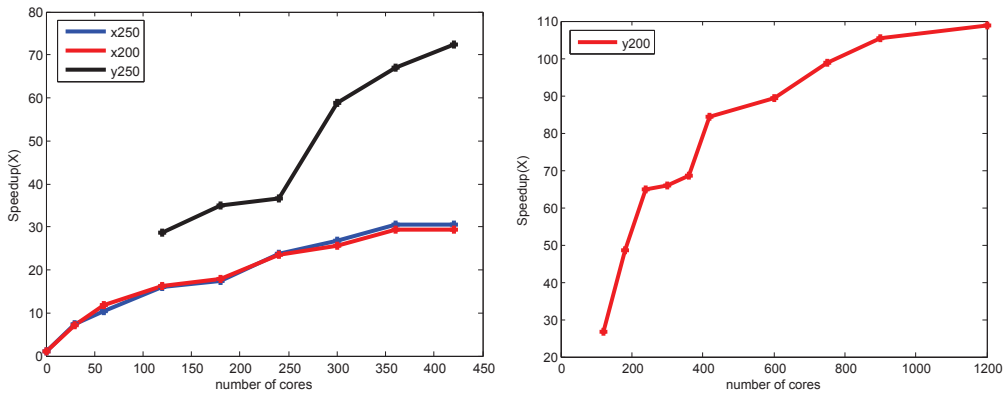
By testing all the benchmarks with the accuracy requirement, we set the stopping criteria to be 10^{-5}.

$$\max_{i=1,\ldots,N} abs(V_{ref_i} - V_i) < 0.01\text{mv} \tag{2.4}$$

In the experiment, the overlapping ratio is defined to be the ratio between half of the overlapping length to the total side length of a subdomain. To achieve a good balance between iteration numbers and matrix size, overlap-

ping ratio is experimentally set to be 0.2.

We test three sets of benchmarks to verify the proposed method, including IBM power grid benchmarks $pg3 - pg6$, $x250, x200, y250, y200$ and four artificially produced benchmarks. The $pg3 - pg6$ benchmarks are small size benchmarks with wire bond packages, while the rest are medium and large size benchmarks with flip chip packages. Simulation results are listed in Tables 2.1, 2.2, and 2.3.



(a) speedup curves for $x250, x200$ and $y250$ (b) speedup curve for $y200$

Figure 2.9: Speedup curves

We test our program on a set of small and the median size IBM power grid benchmarks, where we see a good speedup over the direct LU solver. The data is shown in Table 2.1 and Table 2.2. The maximum speedup of SuperLU_Dist in [21] is 25X with 128 cores, while we achieve over 88X speedup with 120 cores, as shown from the data of $pg3$ in Table 2.1. The largest speedup is over 110X over the direct LU solver, and is over 25X over the sequential version of the proposed method. The speedup curve of benchmarks in Table 2.2 is shown in Figure 2.9.

Table 2.3 shows that the proposed method successfully finishes the simulation of large scale power grids in a short time, while direct LU cannot handle the problem. This is because the memory requirement of solving the problem with LU exceeds the system memory limit. As far as we know, it is the first time that power grid containing 192 M nodes is solved within 5 minutes.

It should be noted that since each processor in the distributed memory system has a small memory size, the power grid needs to be partitioned into many small subdomains so that each subdomain can be fully loaded in

Table 2.1: Simulation results of small benchmarks

ckt	$N(K)$	C	$t_c(s)$	$t_y(s)$	P	$t_p(s)$	$SP(X)$
pg3	851	3	30.9	17.8	60	0.46	67.2
					120	0.35	88.3
					180	0.31	99.7
pg4	953	2	40.63	30.8	60	1.7	23.9
					80	1.35	30.1
					120	1.03	40.5
					180	0.76	53.5
pg5	1070	5	6.59	12.1	60	1.17	5.6
					120	0.62	10.6
					180	0.46	14.3
pg6	1670	2	9.95	19.82	40	1.42	7
					60	1.11	8.9
					120	0.81	12.3
					180	0.74	13.4

N - number of nodes in the grid; C - number of components in the grid; t_c - runtime of UMFPACK; t_y - optimal runtime of a sequential domain decomposition method [20]; P - number of processors; t_p - runtime of proposed method; SP - speedup of proposed method over UMFPACK: $SP = t_p/t_c$.

memory. Each processor works on one subdomain. For large-scale benchmarks, with sufficient processor resource, networks can be solved in parallel, such as $netlist\_32M, netlist\_72M$ and $netlist\_108M$ in Table 2.3. Otherwise, those networks should be solved without the grouping technique, such as $netlist\_144M$ and $netlist\_192M$.

2.4 Summary

We present an efficient parallel geometrical ASM for power grid DC analysis. Based on the distributed memory system, we develop a new data storage method, which efficiently overcome the memory bottleneck of traditional parallel implementations. The largest power grid size that can be solved is not limited by the memory of a single processor. We propose an efficient method to only exchange boundary information of the subdomains. Mini-

Table 2.2: Simulation results of medium benchmarks

ckt	$N(K)$	C	$t_c(s)$	$t_y(s)$	P	$t_p(s)$	$SP(X)$
x250	3782	3	47.14	25.29	120	2.7	17.5
					360	1.58	29.8
x200	5910	3	83.58	53.3	120	4.74	17.6
					360	2.91	28.7
y250	6727	3	1221.5	377.8	120	49.44	30.2
					360	18.01	67.8
y200	10513	3	2710.3	626.6	120	88.64	30.5
					360	41.6	65.15
					1200	24.48	110.7

Table 2.3: Simulation results of large benchmarks

ckt	$N(M)$	C	P	$t_p(s)$
netlist_32M	32	2	500	15.54
netlist_72M	72	2	500	53.13
netlist_108M	108	2	1200	47.48
netlist_144M	144	2	1200	62.02
netlist_192M	192	2	1200	256.91

mum communication overhead is achieved. Besides, processors are grouped into clusters to introduce more parallelism. Experimental results show that more than 110X speedup is gained over a state-of-art LU solver. It is also the first time reported that a power grid containing 192 M nodes can be solved within five minutes.

CHAPTER 3

PGT_SOLVER: AN EFFICIENT SOLVER FOR POWER GRID TRANSIENT ANALYSIS

In transient analysis, the power grid is modeled as a RLC network, which is shown in Figure 3.1. After utilizing equivalent companion models of capacitance and inductance, performing power grid transient analysis is equivalent of solving linear systems at each time step. Transient analysis is more accurate than DC analysis, but is also more time consuming. As a result, it is more urgent to reduce the runtime of transient analysis.

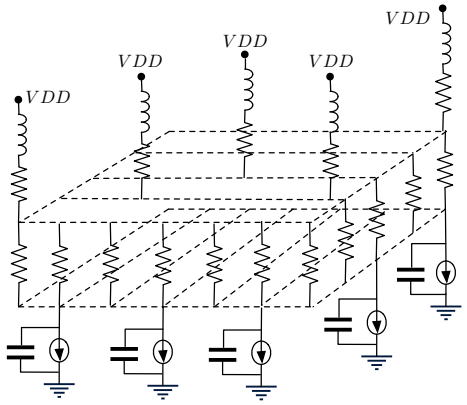


Figure 3.1: Power grid model for transient analysis

As mentioned in Chapter 1, DC analysis and transient analysis have different runtime bottlenecks. Methods that are effective in reducing the runtime of DC analysis may not decrease the runtime of transient analysis. As a result, we need to develop a new method to reduce the runtime of transient analysis.

In this chapter, we present an efficient solver for power grid transient analysis: PGT_SOLVER. It is based on the direct method, and is developed on a shared memory system. A global conductance matrix is built and factorized twice with the Cholesky method: one for DC analysis, the other for the first time step in transient analysis. The decomposed matrix is then reused in the time steps. Forward and backward substitutions are performed

once per time step. Techniques such as sparse vector and solution mapping are proposed to speed up the forward and backward substitutions. Multiple threads are utilized to further reduce the runtime. Each thread works on one network of the power grid.

PGT_SOLVER is developed for "TAU 2012 Power Grid Simulation Contest". It was tested on a set of IBM benchmarks [22]. Criteria such as accuracy, runtime and memory are utilized to evaluate the performance of the solver. It won the 1st place out of eight teams participating the contest. It is fast at getting solutions without introducing any error. The memory consumption is also very affordable.

In the rest of this chapter, Section 3.1 discusses companion models of capacitance and inductance. Section 3.2 presents detail techniques of the solver. Experimental results are illustrated in Section 3.3. The summary is given in Section 3.4.

3.1 Equivalent companion models of capacitance and inductance

There are three equivalent companion models for the capacitance and inductance: Forward Euler (FE), Backward Euler (BE) and Trapezoidal model (TR), which are shown in Figure 3.2. Different models affect whether the system conductance matrix is symmetrical positive definite (SPD) or not. If the matrix is SPD, we can decompose the matrix with the Cholesky method or the LU method. Cholesky decomposition is superior than LU decomposition. It is not only on average 2 times faster, but it also saves about half the memory for storing the decomposed matrix. As a result, it is very important to explore the companion models of these two devices.

In the three companion models, there are two forms of BE and TR models: Norton and Thevenin. In FE model or Thevenin form of BE and TR models, an equivalent voltage source across two nodes is introduced. With any of these models, the conductance matrix is not SPD. This can be easily proved as follows.

Assume the newly introduced voltage source is between node i and node j. Suppose there are N nodes in the grid. Based on modified nodal analysis (MNA), the matrix after stamping the voltage source is shown in Figure 3.3.

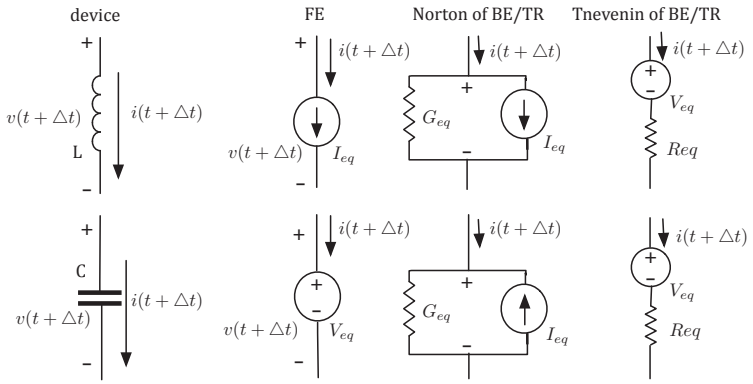


Figure 3.2: Companion models of capacitance and inductance

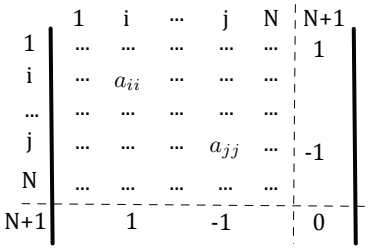


Figure 3.3: Conductance matrix with Norton form of BE/TR companion models

The newly introduced voltage source leads to a "0" diagonal element in the matrix. Different from the matrix for DC, this "0" diagonal element cannot be removed with matrix deformation. If the matrix is SPD, it should have all positive diagonal elements. This necessary condition is violated with the "0" diagonal element generated by the voltage source in the companion models. As a result, the matrix generated by the FE model, Thevenin form of BE/TR model is not SPD.

Meanwhile, with the Norton form of BE/TR companion models for both capacitance and inductance, only resistors and currents are introduced. Without a voltage source across two nodes, the conductance matrix is SPD.

3.2 Methodology

3.2.1 Sparse vector method

With the companion models, performing power grid transient analysis is equivalent to solving a linear system ($GV = I$) at each time step. There are two stages in solving a linear system: the matrix decomposition stage and the substitution stage. The substitution stage includes forward substitution (FS) and backward substitution (BS).

It is well known that the sparsity feature of the matrix greatly reduces computational intensity of matrix decomposition. Meanwhile, sparse vectors can also reduce the computational intensity of the substitutions. This characteristic has not yet been utilized in power grid analysis.

In 1989, Tinney, Brandwajn and Chan [23] first proposed the sparse vector method for a linear system. The ideas of the sparse vector method are illustrated here with a simple linear system of six nodes, where we are solving $GV = I$.

Assume we build the system conductance matrix G and decompose it with the Cholesky method, which is shown in equation (3.1). Equations of forward and backward substitutions are shown in equation (3.2) and equation (3.3). V and I are the voltage and current vectors. L and L' are the decomposed lower and upper triangular matrices.

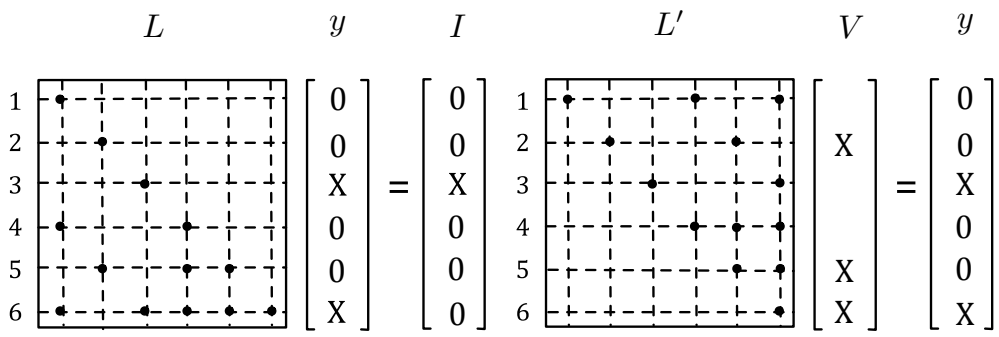
$$G = LL' \qquad (Matrix\ decomposition) \tag{3.1}$$

$$Ly = I \qquad (Forward\ substitution) \tag{3.2}$$

$$L'V = y \qquad (Backward\ substitution) \tag{3.3}$$

In the forward substitution, if the right-hand vector I is sparse, the solution vector y is also sparse. For example, Figure 3.4(a) shows that only I_3 is non-zero, while all other elements of I are zeros. With the non-zeros of L being marked as the dots in Figure 3.4(a), only y_3 and y_6 are non-zeros. If we know beforehand where the non-zeros are, we only need to find the values of the non-zeros instead of solving the whole vector.

If we are only interested in knowing the solutions of part of the nodes, saving can also be achieved in backward substitution, For example, in Fig-

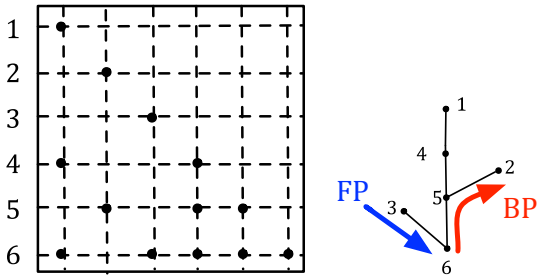


(a) forward solving (b) backward solving

Figure 3.4: Decomposed matrix and path graph

ure 3.4(b), if we are only interested in knowing the value of V_2, then we only need to solve V_6, V_5 and V_2. There is no need to solve other values of V.

The above ideas are realized through building the path graph from L. We build the path graph by creating an undirected edge between each diagonal element of L and its first non-zero element in the same column. In the above example, L with its corresponding path graph are shown in Figure 3.5.



(a) decomposed L (b) path graph

Figure 3.5: Decomposed matrix and path graph

With the path graph and locations of non-zeros or the interested nodes in vector I and V, we record the nodes that need to be solved into two paths, which are defined as "forward path (FP)" and "backward path (BP)". The two paths of the above example are shown in equation (3.4). They are also presented as the "FP" and "BP" in Figure 3.5(b).

$$\begin{aligned} &\{3,6\} && (FS) \\ &\{6,5,2\} && (BS) \end{aligned} \tag{3.4}$$

With the sparse vector method, only partial nodes of the vectors are required to be solved. In order to make the method effective for power grid transient analysis, we need to explore the sparsity of I and the number of interested nodes in V.

First, I is very sparse. The number of non-zeros in I depends on the number of current sources in the grid. Because nodes connecting current sources only occupy a very small ratio of the grid, I is very sparse. Second, the number of nodes in V that need to be solved is also limited. We are only interested in knowing voltage values of the tens of observation nodes in the netlist file. In order to solve these nodes, we have to solve the voltage values of the nodes around capacitances and inductances. They are used to calculate equivalent parameters of the two devices, which are shown as $v(t)$ and $i(t)$ at equation (3.5). Because the capacitance and inductance only occupy a limited part of the grid, the total number of nodes that we need to solve in V is limited.

$$\begin{aligned} L : G_{eq} = \frac{\triangle t}{2L} \quad I_{eq} = i(t) + \frac{\triangle t v(t)}{2L} \\ C : G_{eq} = \frac{2C}{\triangle t} \quad I_{eq} = i(t) + \frac{2C v(t)}{\triangle t} \end{aligned} \tag{3.5}$$

Based on above analysis, we can utilize the sparse vector method to accelerate the forward and backward substitutions. We start by establishing the forward and backward paths, and reuse them along all the time steps.

The process of building forward paths is illustrated in Algorithm 2. Nodes are sorted in ascending order. The backward path can be established in the same way, where the nodes are sorted in descending order.

Algorithm 2: Building forward path

//Build first path segment S_0:

For 1^{st} non-zero index in b, add descendant nodes into S_0 until current node has no descendant in the path graph.

// build p^{th} path segment

p=0; $SS = \emptyset$.

for $i \in$ non-zero indexes of b **do**

  j = i;

  // build a new path segment

  If j^{th} row in L has non-zero and is unvisited

    p = p+1; $S_{p+1} = \emptyset$;

    While n_j has descendant node and the node is unvisited

      $S_{p+1} = S_{p+1} \cup n_j;\ j = j + 1;$

    // include this path segment into the global list

    $SS = SS \cup S_{p+1}$.

end for

sort SS in ascending order.

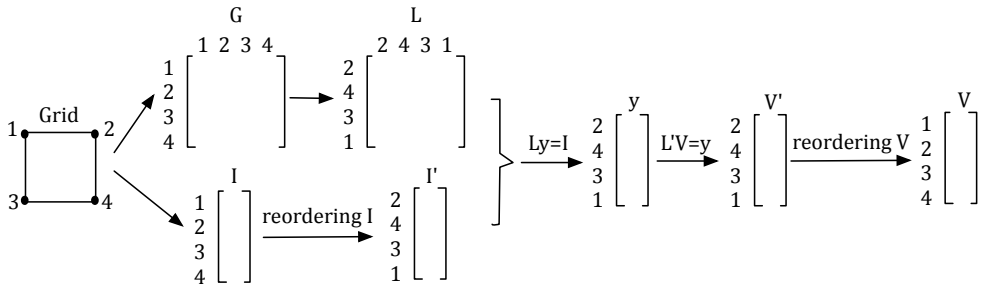
forward_path $\leftarrow$ insert each node in SS into S_0.

3.2.2 Solution mapping

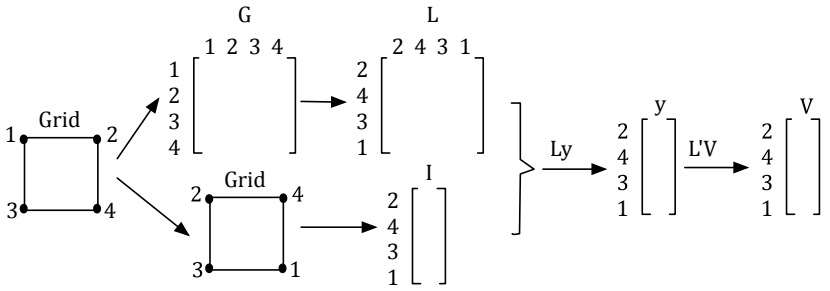
In matrix decomposition, reordering is always utilized to reduce fill-ins. However, this saving brings some extra cost in the forward and backward substitutions. At each time step, the substitutions involves two reordering operations, which are shown in Figure 3.6.

Figure 3.6(a) starts with a grid containing four nodes. The numbers on the nodes show the indexes of the nodes. Matrix G and vector I are formed with the nodes' indexes. Because of the reordering in matrix factorization, the decomposed matrix L has different indexes to those of the nodes. In order to get the correct solutions, we have to perform reordering operations to vector I and V.

The above reordering processes happen at each time step. For the thousands of time steps in transient analysis, this cost cannot be ignored. In this dissertation, we propose a solution mapping technique to avoid the reordering operations.



(a) solving with reordering



(b) solution mapping

Figure 3.6: Solving with reordering and smart mapping

As shown in Figure 3.6(b), after factorizing G, we renew the indexes of nodes in the grid. Current vector I is built with the new indexes of nodes in the grid. With this operation, we can perform the substitutions without reordering. The index renewing process is performed only once in the first time step and is reused at all time steps.

3.2.3 Memorized supernode solving

The supernode technique has been wisely utilized in matrix operations. Instead of performing operations on a single column or row, performing operations on a supernode makes the memory fetch and store more efficiently.

We implement supernodal solving at forward and backward substitutions. After factorization of conductance matrix, limited types of supernodes are located by judging the number of columns that have the same features. Different functions are then utilized for these supernodes in forward and backward substitutions.

For multiple time steps of substitutions, instead of locating the supernodes at each time step, we memorize the supernodes with a simple data structure.

After matrix decomposition, we create an array, where each element of the array records the number of columns of each supernode. Successive elements record data of successive supernodes. From the second time step, by accessing the elements of the array, we avoid judging operations. Functions can be directly called for different types of supernodes.

3.2.4 Multiple threads

In this section, we discuss the parallelization of the above method with multiple threads.

We first consider parallelization of multiple time steps. More specifically, we want to parallelize the substitutions. However, it is hard to bring some speedup for this parallelization. First, the sequential substitutions are very fast. For example, for a power grid with 3 million nodes, substitutions cost less than 1 second per time step. More importantly, the decomposed matrix is sparse and irregular, which makes the parallelization ineffective.

We also tried to parallelize matrix decomposition. Only part of the matrix operations in the decomposition process can be parallelized. Due to the sparsity and irregular structure of the decomposed matrix, the effects can only been seen for a large size matrix, where there are enough tasks for multiple threads to bring some speedup. Furthermore, because of thousands of time steps, matrix decomposition only occupies a limited ratio in runtime. In this case, even if there is some parallelization in the decomposition stage, the speedup for the whole analysis may be insignificant. We performed a 1000 time step transient analysis on a power grid of 3 million nodes. Matrix factorization occupies less than 40% of the total simulation time. With our implementation of parallelization in matrix decomposition, we only observed less than 20% decrease in runtime. With this test case, the maximum speedup that can be achieved by only parallelizing matrix decomposition is less than 2X.

Based on above consideration, we changed our strategy of parallelization. We noticed that there are several networks in the power grid which are usually treated as disjoint, such as VDD, GND and so on. These networks can be solved in parallel. We utilize multiple threads to parallelize the program, where each thread works on one network. If the sizes of the networks are

similar, the number of speedups is almost identical to the number of networks in the grid. Moreover, with this strategy, there is very little effort to modify the program. We parallelize the program by adding only about 2-3 lines. Considering the little workload and the effect of speedup, our strategy for parallelization is efficient to accelerate the analysis.

3.3 Experimental results

The proposed method is developed in C++ and tested with several industrial benchmarks from [24]. The hardware platform is a shared memory system with 32 G RAM and 16 threads. The CPU frequency is 3.33 GHz.

Table 3.1 shows the effect of the sparse vector method on *ibmpg*1*t* to *ibmpg*3*t*. By observing the original number of nodes and the number of nodes in the forward and backward paths, we can see that the sparse vector method reduces the number of the nodes need to be solved by 20%.

Table 3.1: Utilization of sparse vector method

ckt	C	N	N_{ffs}	$r_1(\%)$	N_{fbs}	$r_2(\%)$
*ibmpg*1*t*	VDD	11.6K	11.4K	97.8	11.3K	96.5
	GND	13.9K	10.7K	76.9	10.7K	76.8
*ibmpg*2*t*	VDD	83.9K	65.1K	77.6	65.1K	77.6
	GND	80.3K	65.7K	81.8	65.6K	81.7
*ibmpg*3*t*	VDD	1707	944	55.3	771	45.1
	GND	240K	230K	95.8	230K	95.8
	VDD2	234K	229K	98.1	229K	97.9

C - number of networks in the grid; N - number of nodes of each network; N_{ffs}, N_{fbs} - number of nodes in forward and backward path; r_1 - N_{ffs}/N; r_2 - N_{fbs}/N.

Data in Table 3.2 shows the runtime with the proposed techniques. Each of the techniques helps reduce runtime. Especially, there is a big time saving when applying the multithread technique. It should be noticed that the three benchmarks are small. For larger benchmarks, the saving in runtime will be more prominent.

The proposed solver is developed for the "TAU 2012 Power Grid Simulation Contest". There are eight teams participating the contest. Programs are evaluated based on 64-bit Intel Xeon CPU with a frequency of 2.27 GHz.

The system has 64 G RAM and 32 threads. Programs are verified through a set of IBM power grid benchmarks in [22], which are generated based on the DC benchmarks in [24]. Accuracy, runtime and memory are the factors used to evaluate the performance of the programs. Each of them $\in$ [0,100]. The lower the score, the better the performance. Detailed criteria can be checked at official website of "TAU 2012 Power Grid Simulation Contest" [22].

Table 3.2: Runtime with different techniques

ckt	T_0	T_1	T_2	T_3	T_4
*ibmpg*1*t*	2.67s	2.03s	2.02s	2.02s	1.6s
*ibmpg*2*t*	29.63s	28.32s	27.41s	27.25s	22.34s
*ibmpg*3*t*	1m56s	1 m44s	1m37s	1m36s	1m15s

T_0 - CHOLMOD solver; T_1 - CHOLMOD + solution mapping; T_2 - Method in T_1 + sparse vector; T_3 - Method in T_2 + memorized supernode; T_4 - Method in T_3 + multithread.

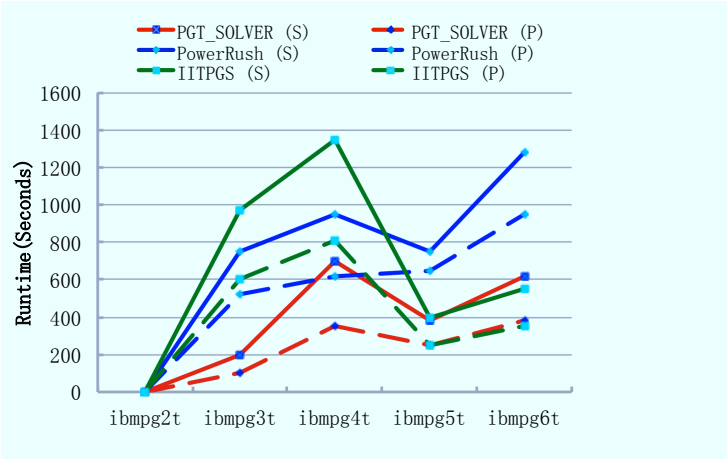
Table 3.3: Scores for top three teams

Team	T_s	T_p	M_s	M_p	E_s	E_p
1*st*	114	67	190	320	0	0
2*nd*	229	169	100	160	49	49
3*rd*	239	150	415	235	49	250

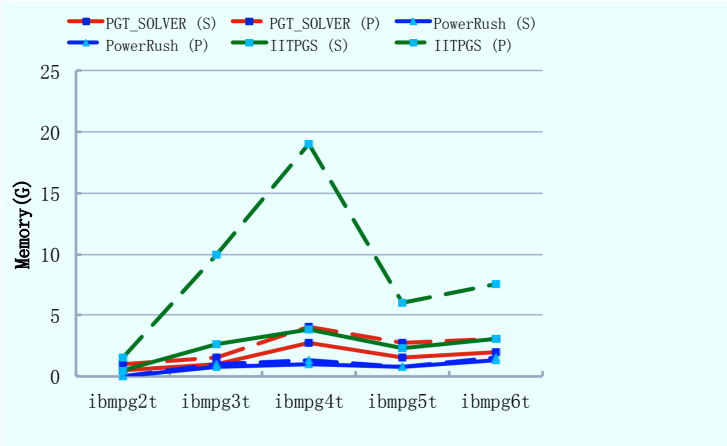
T_s/T_p - runtime scores in sequential and parallel modes; M_s/M_p - memory scores in sequential and parallel modes; E_s/E_p - error scores in sequential and parallel modes.

The proposed solver won the first place in the contest. Table 3.3 lists the scores of the top three teams. The proposed method is the only one that does not introduce any error in the final solutions. Figure 3.7 shows runtime and memory consumption under sequential (S) and parallel (P) modes of the top three teams. "PGT_SOLVER", "$PowerRush$" [25] and "$IITPGS$" [26] refer to the methods of the first, second and third winning teams of the contest. Because the platform utilized in the contest has slower CPU frequency, runtime for *ibmpg*2*t* and *ibmpg*3*t* is higher than that of Table 3.2.

Figure 3.7 shows that the proposed method achieves the fastest runtime in both sequential and parallel modes. Generally, our PGT_SOLVER saves about 50% of runtime compared to $PowerRush$, which is the second-place



(a) runtime of top three teams



(b) memory of top three teams

Figure 3.7: Runtime and memory consumption of top three teams in TAU 2012 conest

winner. At the same time, memory consumption of PGT_SOLVER is only a little higher than $PowerRush$, which has the best memory consumption of the three teams.

3.4 Summary

In this chapter, we propose PGT_SOLVER, which is an efficient solver for power grid transient analysis. Trapezoidal models of capacitance and inductance are utilized to generate symmetric positive definite (SPD) system conductance matrix. Techniques such as sparse vector and solution mapping are developed to accelerate forward and backward substitutions. Multiple threads are utilized to further reduce the runtime of the simulation. Being tested on a set of industrial benchmarks, PGT_SOLVER won the first place of the eight teams that participated in "TAU 2012 Power Grid Simulation Contest". PGT_SOLVER is the fastest under both sequential and parallel modes. With reasonable memory consumption, the solver only costs half the runtime over the second place winner of the contest.

CHAPTER 4

EFFICIENT PARALLEL SOLVER FOR POWER GRID TRANSIENT ANALYSIS

In Chapter 3, we propose *PGT_SOLVER* to accelerate power grid transient analysis. It is based on a shared memory system. It achieves the fastest runtime of the top three methods in the "TAU 2012 power grid simulation contest". Transient analysis is difficult to parallelize. For example, in the TAU 2012 contest, although different techniques are applied to parallel the sequential methods, less than 2X speedup is achieved comparing to the sequential method. Besides, the methods cannot fully utilize the parallel resources. With 16 threads, the best effective number of threads is only 4.3. Based on these considerations, it is necessary to develop a more efficient method to fully utilize the parallel resources and accelerate the simulation.

In Chapter 2, we propose an efficient parallel domain decomposition method [27] to perform power grid DC simulation. Over 100X speedup is achieved compared to a state-of-art direct solver. By utilizing more than 1000 processors, a large size power grid can be quickly solved. Although the method achieves good performance for DC analysis, it may not be the same case for transient analysis. In order to achieve good performance, special considerations need to be made for transient analysis, such as power grid partitioning.

Combining the advantages of the parallel method for DC analysis and *PGT_SOLVER*, we propose an efficient parallel solver for power grid transient analysis. It is developed on a distributed memory system. An effective partition method is proposed to divide the power grid into a set of subdomains. The parallel flow of DC analysis and techniques of *PGT_SOLVER* are utilized to accelerate the transient simulation. Solving large size power grids requires more memory than the system's memory size. For this case, we perform the parallel process in multiple steps. The processors solve the subdomains with partial sequential order, which reduces the memory usage of the program.

In the rest of this chapter, Section 4.1 shows the flow of the proposed

method. Section 4.2 presents the partitioning of the power grid. Experimental results and summary are illustrated in Section 4.3 and Section 4.4.

4.1 Flow of the proposed method

The flow of the proposed method is shown in Figure 4.1. We first utilize a single processor to explore the partitioning of the power grid. With the partitioning, the power grid is divided into a set of subdomains. Net connections of each subdomain are extracted and stored in a file. Each processor handles one subdomain. All the processors parse the files and factorize the matrices of all the subdomains. The factorized matrices are reutilized for all the time steps.

As shown in Figure 4.1, each time step is solved iteratively. In each iteration, forward and backward substitutions are performed to solve the subdomains. Advanced techniques of PGT_SOLVER [28] are utilized to accelerate the substitutions. In the end of each iteration, the processors exchange the voltage values of boundary nodes with the efficient communication method of DC analysis [27].

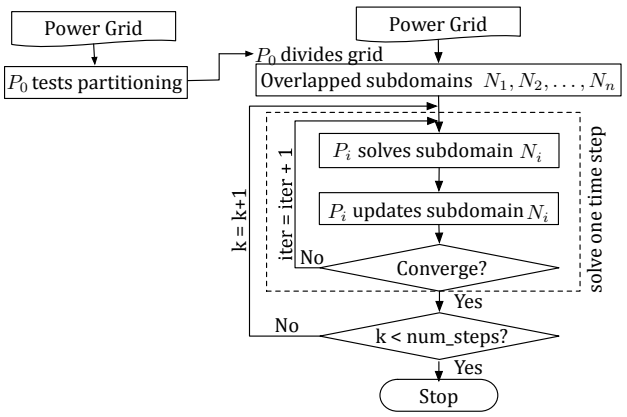


Figure 4.1: Flow of the proposed method

In the flow, by default, the subdomains are solved with the direct method. If the power grid is in large size, due to the limited memory size of each processor, the grid has to be divided into a lot of subdomains to be solved. This requires a lot of processors, which may be over the limit of the parallel system.

To address this issue, we have to reduce the memory consumption for the simulation, so that the power grid can be handled with fewer processors. Instead of having all the processors build and factorize the matrices of the subdomains simultaneously, we decided to execute the process in several steps. We divide the processors into several groups, where each group of processors executes the process simultaneously. Processors of different groups execute the process sequentially. By having fewer processors working in parallel, the memory requirement is alleviated.

4.2 Partitioning of the power grid

Partitioning is an important step of the flow. It affects both the workload of computation and communication. In order to achieve good performance of transient analysis, it is necessary to discuss the partitioning of the power grid.

4.2.1 1D partitioning versus 2D partitioning

Similar to [27], we only consider partitioning along horizontal and vertical directions. Possible partitions include one-dimensional (1D) partitioning and two-dimensional (2D) partitioning. As DC analysis usually starts with an initial solution that is far away from the final solution, it usually takes a lot of iterations to converge. In this case, 2D partitioning is beneficial as it effectively reduces the number of iterations. For example, in the DC analysis [27], 2D partitioning reduces the number of iterations from hundreds to tens. Faster DC analysis is achieved by performing 2D partitioning.

Different from DC analysis, each transient step begins with good initial voltage values. It only requires tens of iterations to converge to a final solution. Due to this feature, the above advantage of 2D partitioning in transient analysis is much weaker. On the other hand, the side effect of 2D partitioning may dominate transient analysis and results in longer runtime. For example, compared to 1D partitioning, 2D partitioning introduces more fill-ins in the decomposed matrices. Figure 4.2 illustrates such a case, where both 1D and 2D partitioning are performed for a regular power grid. A top view of the power grid is illustrated in the figure. The red dotted lines show the

partitioning.

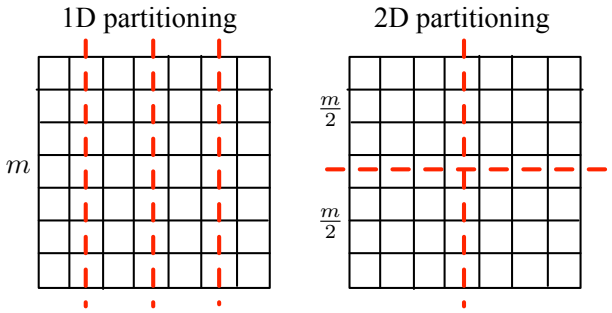


Figure 4.2: Partitioning of power grid

Assume there are m nodes in each column of the grid. In Figure 4.2, each subdomain of 1D partitioning has two columns of nodes. Each column has m nodes. Each subdomain of 2D partitioning has four columns of nodes. Each column contains $\frac{m}{2}$ nodes.

With the above assumption, the system conductance matrices of one subdomain of 1D and 2D partitioning are shown in Figure 4.3. The black solid lines show the locations of non-zeros. After matrix factorization, a set of fill-ins is introduced, which are shown as the red solid lines. The numbers of fill-ins of the two cases are as follows:

$$N_{(fill-in)\_1D} : m - 1$$
$$N_{(fill-in)\_2D} : (\frac{m}{2} - 1) \times 3 = \frac{3m}{2} - 3$$

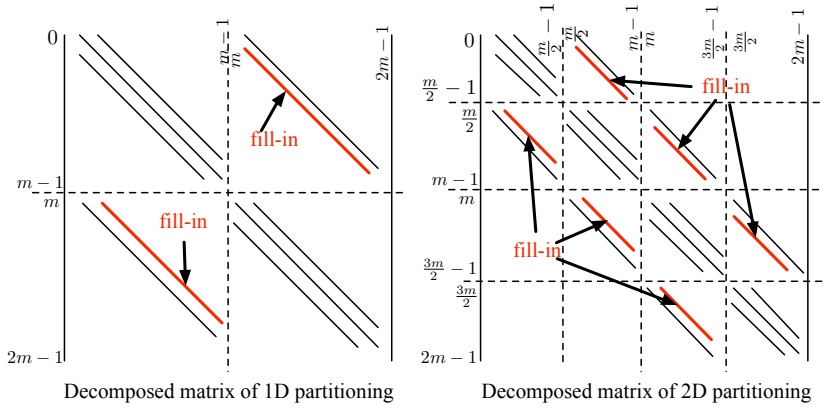


Figure 4.3: Decomposed matrices of 1D and 2D partitioning

$$N_{(fill-in)\_2D} - N_{(fill-in)\_1D} = \frac{m}{2} - 2 > 0 \;\; if(m > 4) \tag{4.1}$$

Because m is usually a big number, 1D partitioning introduces fewer fill-ins than that of 2D partitioning. As a result, the forward and backward substitutions with 1D partitioning are faster than that of 2D partitioning. Considering that the substitutions are performed at each iteration of all the time steps, simulation with 1D partitioning can be much faster than that with 2D partitioning.

4.2.2 Partitioning direction

Based on the above analysis, 1D partitioning should be utilized to reach better performance for transient analysis. Depending on the feature of the power grid, such as net and resistance distributions, the direction of partitioning may also affect the transient analysis performance.

4.2.2.1 Considering nets' distribution

As mentioned before, power grid structure can be irregular. The number of nets along the vertical and horizontal directions may be very different. Figure 4.4 shows such a case. There are more nets along the vertical direction than the horizontal direction. Resistance values of all the nets are the same. In this case, cutting the grid vertically results in more boundary nets than cutting it horizontally. In order to reduce the overhead of communication, it is preferable to perform horizontal partitioning.

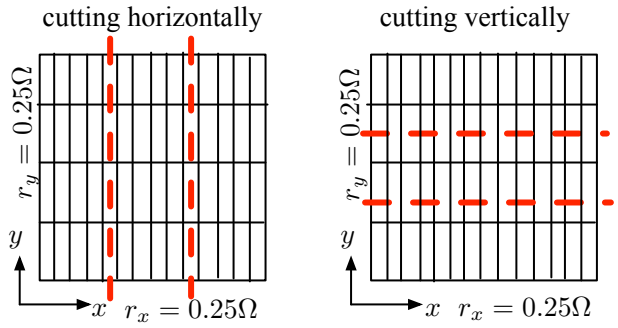


Figure 4.4: Power grid with different number of nets along x and y directions

4.2.2.2 Considering resistance values

Figure 4.5 shows a power grid with uniform net distribution but with different resistance values along horizontal and vertical directions. Vertical resistance values are smaller than horizontal resistance values. In this case, cutting the grid vertically exposes the resistance nets with smaller values to the iterative process. Compared to cutting the grid horizontally, it results in a more "ill-conditioned" system, which requires more iterations to converge to the final solutions. As a result, it is preferable to cut the nets with bigger resistance values.

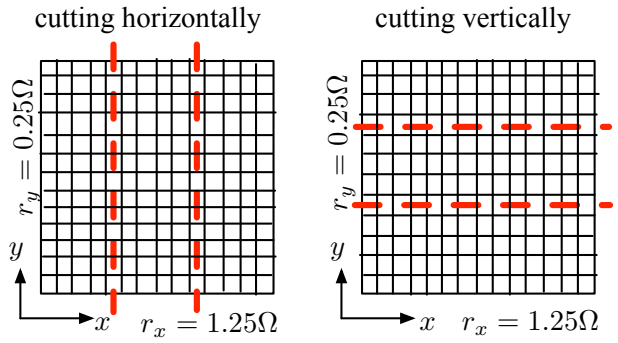


Figure 4.5: Power grid with different resistance values along x and y directions

4.2.2.3 Partitioning method

Considering both the nets' distribution and the resistance values of the power grid, we develop an efficient sequential method to obtain the partitioning direction. Pseudocode of the method is shown in Algorithm 3, where $TH1$ and $TH2$ are threshold values. $TH1/TH2 \in [0.0 - 1.0]$.

The flow of the algorithm is as follows. We scan all the resistor nets of the grid and calculate the total number of nets and total resistance values along vertical and horizontal directions. Nets that are not strictly horizontal or vertical are counted in both directions. After scanning the nets, we obtain the number of nets and accumulated resistance values along the two directions. We first check the number of nets along the two directions. If the number of nets along one direction is obviously less than the other direction, then the partitioning should be performed along this direction. If the numbers of nets

Algorithm 3: Algorithm of partitioning

```
1  begin
2      // scan the net connections
3      for i=1,...size(resistor_net_set) do
4          num_nets_x++   if net is not vertical net;
5          num_nets_y++   if net is not horizontal net;
6          R_x += net_value   if net is not vertical net;
7          R_y += net_value   if net is not horizontal net;
8      end
9      ratio_nets = min(num_nets_x/num_nets_y, num_nets_y/num_nets_x);
10     if ratio_nets > TH1 then
11         ratio_r = min(R_x/R_y, R_y/R_x);
12         if ratio_r > TH2 then
13             cutting the grid either along horizontal or vertical direction;
14         else if R_x > R_y then
15             cutting the grid horizontally (e.g., 32 × 1(x × y));
16         else
17             cutting the grid vertically (e.g., 1 × 32(x × y));
18         end
19     else if num_nets_x > num_net_y then
20         cutting the grid vertically (e.g., 1 × 32(x × y));
21     else
22         cutting the grid horizontally (e.g., 32 × 1(x × y));
23     end
24 end
```

along the two directions are similar, we further check the total number of resistance values along the two directions. If one direction has a much bigger resistance value than the other, the grid should be cut along this direction.

4.3 Experimental results

The proposed method is developed with C++. Numerical package $CHOLMOD$ is utilized to perform matrix factorization. IBM benchmarks [24] and a set of artificial benchmarks are utilized to test the performance of the proposed method. Each test case has 1000 fixed time steps. We only measure the runtime to solve the grids. Runtime of parsing is not included.

The program is implemented on a distributed memory system. The parallel platform consists of approximately 200 cluster compute nodes, each with two 2.67 GHz Intel Xeon hex-core processors and 24 GB of RAM. Each processor has about 1.2 GB memory. The message passing package MPICH2 is used.

4.3.1 Accuracy of the proposed method

We first compare the solutions of the proposed method to that of SPICE. IBM benchmark *ibmpg1t* [24] is utilized as the test case. In the proposed method, the grid is divided into four subdomains. The overlapping ratio is the same as [27]. With the convergence threshold set as $\epsilon = 1e-5$, the voltage values of a node are shown in Figure 4.6. We can see that the voltage values of the proposed method are almost the same as the values for SPICE.

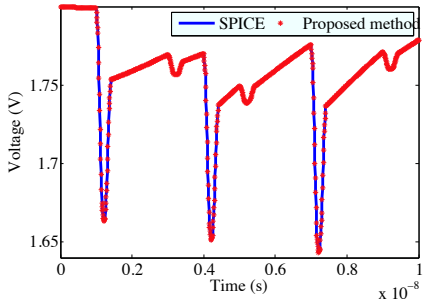


Figure 4.6: Voltage values of a node of power grid

4.3.2 Effect of partitioning

Next, we verify the effectiveness of the proposed partitioning method, including both the 1D partitioning and the partitioning direction. Artificial power grids $C1, C2$ and $C3$ are generated and utilized as the test cases.

There are 100 K nodes in $C1$ and $C3$. $C2$ has 500 K nodes. Assume the number of nets along x and y directions are N_x and N_y. Also assume the total resistance values along x and y directions are R_x and R_y. For the three cases, the number of nets and resistance values along the horizontal and vertical directions have the following relationships:

$$\begin{aligned} C1&: N_x = 0.2N_y;\ R_x = R_y; \\ C2&: N_x = N_y;\ R_x = 0.2R_y; \\ C3&: N_x = 0.2N_y;\ R_x = 0.2R_y \end{aligned} \tag{4.2}$$

We test the proposed partitioning algorithm to obtain the direction of the partitioning. In Algorithm 3, we set $TH1 = TH2 = 0.8$. With this setting,

Table 4.1: Simulation of 1D and 2D partitioning

Ckt	Par	$T_p(s)$	Ckt	Par	$T_p(s)$	Ckt	Par	$T_p(s)$
	3x3	233.9		3x3	360.5		3x3	228.1
$C1$	**9x1**	149.8	$C2$	9x1	213.7	$C3$	**9x1**	148.8
	1x9	157.2		**1x9**	196.7		1x9	156.4

Par - partition of power grid ($x \times y$); T_p - runtime of proposed method.

Table 4.2: Simulation of *ibmpg4t*

Par	$T_p(s)$	$SP(X)$	Par	$T_p(s)$	$SP(X)$
1x36	124.1	2.7	1x204	12.9	26
1x72	57.7	5.9	1x360	11.1	30.5
1x144	28.7	11.8	1x480	12.3	27.5

$SP(X)$ - $338.3s/T_p$.

Algorithm 3 suggests the following partitioning directions:

$$\begin{aligned} &C1\text{: } \textit{horizontal partitioning}, \textit{e.g.}, \ 32 \times 1(x \times y) \\ &C2\text{: } \textit{vertical partitioning}, \textit{e.g.}, \ 1 \times 32(x \times y) \\ &C3\text{: } \textit{horizontal partitioning}, \textit{e.g.}, \ 32 \times 1(x \times y) \end{aligned} \tag{4.3}$$

In order to verify the algorithm, we perform 1D partitioning along both x and y directions. 2D partitioning is also listed as a comparison. Simulation results with a fixed number of processors are shown in Table 4.1.

From Table 4.1, we can see that the simulation with 1D partitioning is much faster than that of 2D partitioning. The runtime difference increases as the test case gets bigger. For example, for $C1$ and $C3$, simulation with 1D partitioning saves around 70 s than that with 2D partitioning. For $C2$, which contains more nodes, simulation with 1D partitioning saves 160 s than that of 2D partitioning.

Table 4.1 also verifies that performance of the 1D partitioning is affected by the direction of the partitioning. The 1D partitions with faster runtime are marked in bold font. They are in accordance with the partitioning algorithm, which are shown in equation (4.3). Comparing the data of $C1, C2$ and $C3$, we can see that the distribution of nets has a bigger effect on the performance than that of resistance values.

4.3.3 Comparing the proposed method with the "TAU 2012" contest

In order to verify the efficiency of the proposed method, we utilize the performance of the top three teams of the "TAU 2012 power grid simulation contest" as a reference. As shown in Figure 3.7(a), less than 2X speedup is reached for all the three teams. In the tests, *ibmpg4t* is one of the most time-consuming benchmarks.

To make a fair comparison, we implement PGT_SOLVER, which is the method of the first-place winner of the contest. Based on the direct method and utilizing the advanced techniques, the method is the fastest sequential solver for power grid transient analysis. *ibmpg4t* is utilized to compare the proposed method and PGT_SOLVER.

Running under sequential mode, PGT_SOLVER costs 338.3s to solve *ibmpg4t*. By applying different partitions, simulation results of the proposed method are shown in Table 4.2. Compared to the advanced sequential method, the proposed method costs much less runtime to solve *ibmpg4t*. For example, with 360 processors, the proposed method only costs 11 s to perform the transient analysis. More then 30X speedup is achieved over PGT_SOLVER. Compared to the 2X speedup of the methods in the "TAU 2012 power grid simulation contest", the proposed method achieves much better performance for power grid transient analysis.

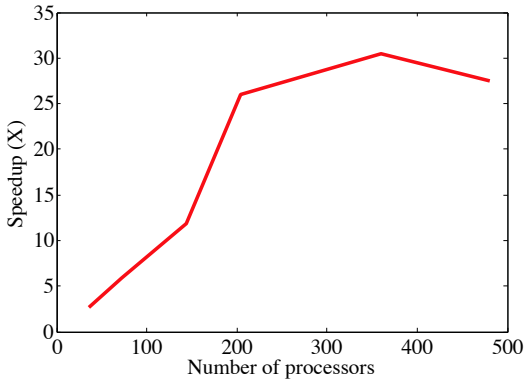


Figure 4.7: Scalability curve of solving *ibmpg4t*

The scalability curve of the simulation is shown in Figure 4.7. Initially, the speedup increases fast with more processors being integrated into the system. However, as the number of processors becomes large, the speedup starts to decrease. This is because the large number of processors requires

Table 4.3: Testing other IBM benchmarks with 1D partitions

Ckt	C	$N(M)$	$T_{seq}(s)$	Par	$T_p(s)$	$SP(X)$
$ibmpg2t$	2	0.08	31	36x1	7.2	4.3
$ibmpg3t$	3	0.5	110.8	204x1	20.5	5.4
$ibmpg5t$	5	0.8	216.8	320x1	19.3	11.2
$ibmpg6t$	2	1.2	451.5	320x1	20	22.6

C - number of independent networks in the test case; N - number of nodes in each network; T_{seq} - runtime of sequential PGT_SOLVER.

Table 4.4: Simulation of medium size test cases

Ckt	T_{seq}(h:m:s)	Par	T_p(h:m:s)	$SP(X)$
ckt_1.5M	00:11:03	144x1	00:00:53	12.5
		204x1	00:00:29	22.8
ckt_3M	3:31:33	1x204	00:03:02	69.7

more iteration to solve the time steps. As the number of processors gets larger, the overhead caused by the extra iterations contributes more and more to the runtime, which finally results in slower runtime.

We also test the proposed method on other IBM power grid benchmarks. The simulation results are listed in Table 4.3. The results verify that the proposed method is efficient in performing parallel power grid transient analysis.

Since the IBM power grid benchmarks are small, we create a few bigger size benchmarks to further test the proposed method. Each benchmark only contains a single network. The simulation results with PGT_SOLVER and the proposed method are listed in Table 4.4. The numbers of nodes of the benchmarks are included in the names of the test cases. The results verify the advantage of the proposed method. For example, for ckt_3M, the proposed method achieves 69X over PGT_SOLVER. With only 204 processors, the proposed method reduces the runtime of from three and a half hours to three minutes.

4.3.4 Simulation of large size power grids

In order to further verify the effectiveness of the proposed method, we generate a set of large size power grid benchmarks and test the proposed method.

Table 4.5: Simulation of large size test cases

Ckt	T_{seq}	Par	T_p(h:m:s)
ckt_4M	NA	204x1	00:01:31
ckt_8M	NA	1x360	00:03:19
ckt_16M	NA	1x360	00:05:03
ckt_32M	NA	200x1	00:06:05
		600x1	00:11:34
ckt_64M	NA	*200x1	01:20:00
		400x1	00:19:24

* solved with partial parallel process

The numbers of nodes are included in the names of the test cases. Each benchmark contains a single network. Simulation results are shown in Table 4.5.

Due to the memory limitation, PGT_SOLVER cannot process these benchmarks. The sequential runtime is marked with "NA". With the proposed method, these power grids are solved very quickly. For example, with 200 processors, ckt_32M, which contains 32 millions of nodes, can be solved in six minutes. Solving ckt_32M with 600 processors costs more runtime than with 200 processors. This verifies the decreasing stage of the scalability curve, which is shown in Figure 4.7.

Due to the limited memory size of each processor, processing a larger power grid requires more processors. For example, in Table 4.5, ckt_32M can be solved with 200 processors, while solving ckt_64M requires 400 processors. The large number of processors may be out of the limit of the users' parallel system.

To address the issue, we decide to solve the subdomains with partial parallelization. Instead of having all the processors build and factorize the matrices of the subdomains simultaneously, we divide the processors into 20 groups. Each group of processors does not start the operations until the previous group finishes the work. Processors of the same group perform the operations in parallel. With the method, ckt_64M are successfully solved by the 200 processors in one hour and twenty minutes. Compared to the original 400 processors that are required to solve the grid, the method greatly reduces the system requirement to solve the large size power grid.

4.4 Summary

We develop an efficient parallel domain decomposition method for power grid transient analysis. We discuss the effect of partitioning and propose an effective method to divide the power grid into a set of subdomains. Advanced techniques are utilized to accelerate the data communication and substitution processes. We propose an effective method to reduce the memory usage of solving large power grids, which makes the proposed method friendly to the users with fewer processors. The proposed method achieves 69X speedup over sequential PGT_SOLVER. Large size power grids are solved very efficiently.

CHAPTER 5

A NOVEL AND EFFICIENT METHOD FOR POWER PAD PLACEMENT OPTIMIZATION

In this chapter, we present a novel and efficient iterative method for power pad placement optimization. It applies for a power grid with flip chip packages. With fixed number of power pads, we focus on optimizing the locations of the pads to reduce static IR-drop values. The power grid is modeled as a two-dimensional resistor network. Only the VDD network is explored.

We develop a novel IR-drop driven method to calculate the new locations of all the pads. Moving the pads into the calculated new locations reduces local IR-drop values. We develop a pad connection graph to move multiple pads to the calculated new locations. After moving the pads, IR-drop values of the power grid are updated with DC analysis.

We verify the effectiveness of the proposed method on a set of benchmarks. Experimental results show that with only several iterations, the proposed method effectively reduces the IR-drop values of the grid.

The rest of this chapter is organized as follows. Section 5.1 illustrates the proposed method. Section 5.2 lists the experimental results of the proposed method versus others. Summary is given in Section 5.3.

5.1 Method

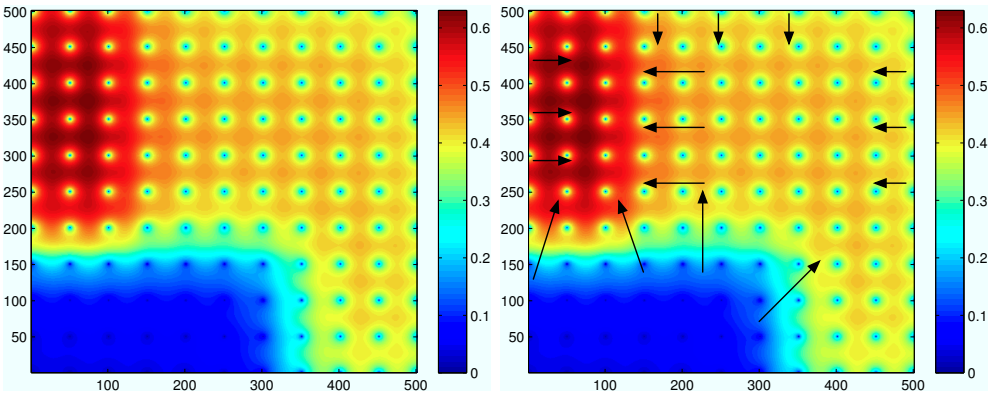
In this section, we introduce the details of the pad placement optimization method, including calculating the new locations of pads, the strategy of moving pads with the pad connection graph and so on.

5.1.1 Calculating new locations of pads

5.1.1.1 IR-drop distribution and control areas of power pads

Figure 5.1(a) shows the IR-drop values of an artificially generated power grid. The numbers are in volts. The grid has 250 K nodes. There are 121 power pads that are uniformly distributed in the grid.

The grid can be divided into three areas: the left-bottom area has the lowest IR-drop values; the top-left area has the highest IR-drop values; the rest of the grid has middle IR-drop values. In order to reduce the IR-drop values, we can see that the lower-bottom area had excessive power pads and we should move these pads toward the left-top and the rest area of the grid. Power pads located in the majority of the middle area should also be moved toward the high IR-drop area. Besides, power pads that are located in the boundaries of the power grid should be moved toward the center of the grid.



(a) IR-drop distribution (b) moving trend of power pads

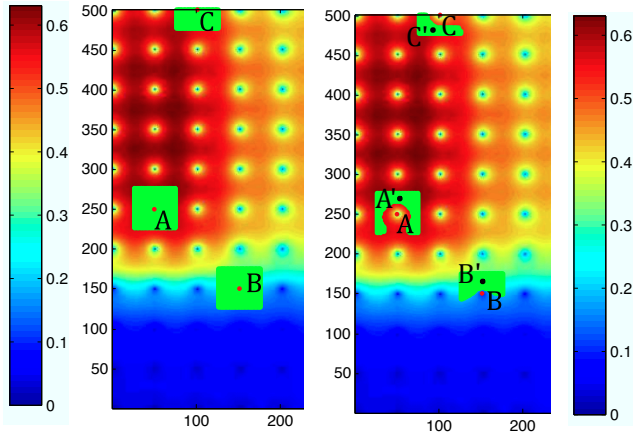
Figure 5.1: IR-drop distribution of a grid and moving trend of pads

We quantify the movement of the power pads by exploring the IR-drop values of local areas around the pads. By comparing the IR-drop values of the nodes in the local areas around the pads, we can calculate the new locations of the pads.

First, we extract a local control area of each pad. Nodes of the control areas are affected most by the corresponding pads. We define the nodes as control nodes of the pads.

To locate the control area of each pad, we start from each node in the grid and locate its geometrically closest pad with Euclidean distance. Because the

power grid is generally in uniform interconnection, the distance between a pad and a node indicates the effect of the pad on the node. After finding the closest pad for all nodes, we map the relationship to the pads and obtain all the control nodes for each pad. These control nodes form the control areas of the pads. For example, Figure 5.2(a) shows pads A, B and C and their control areas, which are marked in green color. We further prune half of the control nodes with lower IR-drop values, so that the rest of nodes in control areas make it clearer to decide the new locations of power pads. The new control areas of the three pads are shown in Figure 5.2(b).



(a) power pads and control areas

(b) pads and modified control areas

Figure 5.2: Power pads and control areas

5.1.1.2 New locations of pads

Assume the candidate locations of power pads are along a coarse grid. Then the new location of a pad is defined as the weighted geometrical center of its control area with a projection to the candidate locations of pads. The weighted geometrical center of each control area is calculated as follows.

We define a weight w_i for each control node i. Weight w_i is in proportional with the nodes' IR-drop values. Nodes with higher IR-drop values have more impact on deciding the new locations of pads. The more nodes with high IR-drop values, the closer the pad's new location to these nodes. Assume the coordinate of a pad is (X, Y). Also assume the coordinate of the i^{th} control node is (x_i, y_i). The new coordinate of the pad is:

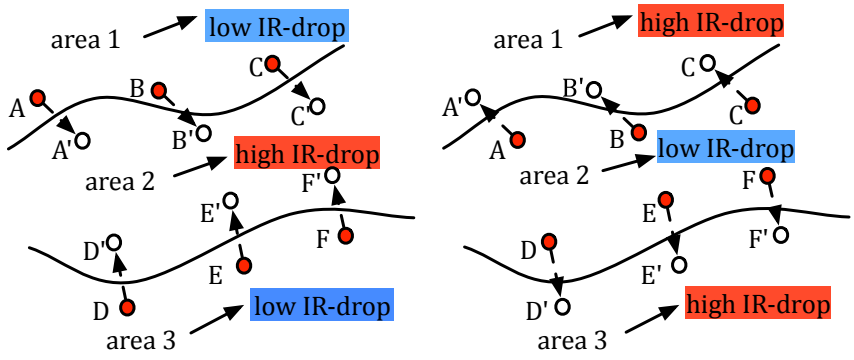
$$(X_{new}, Y_{new}) = (\frac{\sum_{i=1}^{K} w_i x_i}{\sum_{i=1}^{K} w_i}, \frac{\sum_{i=1}^{K} w_i y_i}{\sum_{i=1}^{K} w_i}) \tag{5.1}$$

In equation (5.1), K is the number of control nodes of the pad and $w_i = 1/v_i$, where v_i is the voltage value of the i^{th} control node.

With the above assumption, the new locations of pads A, B and C are A', B' and C', which are shown in Figure 5.2(b). We can see that pad A is moving toward the higher IR-drop area, pad B is escaping from low IR-drop areas and moving to high IR-drop areas. Pad C is moving from the boundary area toward the center of the power grid. These calculated new locations of power pads are in accordance of the moving trend of the pads. There is a high potential to reduce the IR-drop values of the grid.

5.1.2 Moving pads with graph constraints

Moving all the pads into the their calculated new locations helps to reduce local IR-drop values but not global IR-drop values. It may even risk turning low IR-drop areas into high IR-drop areas, which is illustrated in Figure 5.3.



(a) IR-drop values before moving pads (b) IR-drop values after moving pads

Figure 5.3: Effect of moving all power pads simultaneously

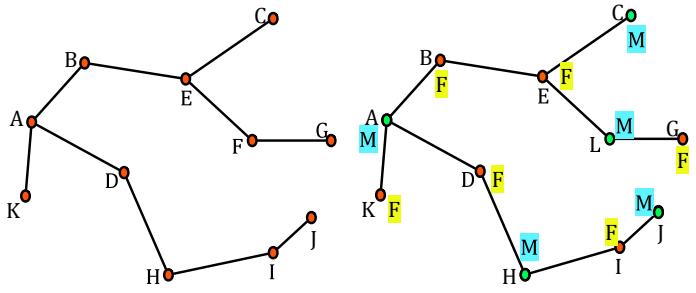
Figure 5.3(a) shows pads $A-F$ and three areas. The middle area has high IR-drop values, while the top and bottom areas have low IR-drop values. Initially, pads $A-F$ are located in the low IR-drop areas. With the above method, the new locations for the pads are $A'-F'$, which are located in the

high IR-drop area. If we move all the pads to the new locations, the top and bottom areas may have high IR-drop values, as shown in Figure 5.3(b). Besides, in next iterations, all the pads tend to go back and forth between the low and high IR-drop areas, thus cannot further reduce the IR-drop values of the grid.

5.1.2.1 Moving pads along pad connection graph

To solve the above problem, once we move a pad, we decide not to move its neighboring pads, so that they can maintain the local IR-drop values. We build a graph to represent the pad connections. Each pad is a vertex in the graph. For each pad, we create an edge between this pad and its geometrical closest neighboring pad. An example graph is shown in Figure 5.4(a).

We start with the pad whose control area has the highest IR-drop values. We move this pad into its calculated new location, and mark it with "Move". If the new location is already occupied by some other pad, we search around the neighboring area of the new location and move the pad to the first available candidate location of the power pad. Neighboring pads of this pad are not moved and are marked with "Fix". We repeat this process until all the pads are marked with "Move" or "Fix". The final graph with all the pads processed is shown in Figure 5.4(b). "M" and "F" represent "Move" and "Fix" respectively.

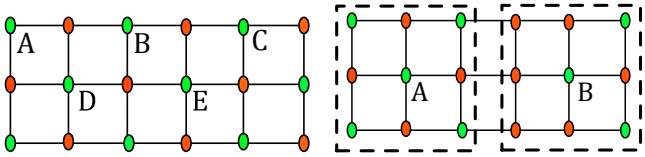


(a) pad connection graph (b) moving pads along graph

Figure 5.4: Moving pads with pad connection graph

5.1.2.2 Number of pads that can be moved per iteration

We further discuss the number of pads that can be moved at each iteration. The number of movable pads depends on the distribution of power pads. Here we only discuss the uniform distribution. Meanwhile, IR-drop distribution decides the moving order of pads, thus affects the number of movable pads. With different IR-drop distributions, we discuss the worst and best cases for the number of movable pads.



(a) best case (b) worst case

Figure 5.5: Number of pads that can be moved

Figure 5.5(a) and Figure 5.5(b) shows the best and worst cases for the number of movable pads. Nodes in green and red represent pads that can and cannot be moved respectively. The best case is shown in Figure 5.5(a), where the movable pads are $A, B, C, D, \ldots$. Every one of the two pads is movable. Assume there are P_1 and P_2 pads along the horizontal and vertical directions. The number of movable pads under the best case can be approximated with equation (5.2):

$$N_M = \frac{1}{2} P_1 \times P_2 \tag{5.2}$$

N_M represents for number of movable pads. $P_1 \times P_2$ is the total number of pads on the grid.

Figure 5.5(b) shows the worst case for the number of movable pads. In this case, pads that are moved first locate at the centers of five-point stencils. There is no overlapping of pads between different stencils. For example, after moving A, the next candidate is pad B, which locates in the center of another five-point stencil. The two stencils do not share any pad, which increases the number of pads that cannot be moved. In Figure 5.5(b), the two center pads of the stencils and six neighboring pads around the stencils are movable. Here we consider on average three neighboring pads around every two stencils are movable, because whether a boundary pad around the stencil is movable affects other boundary pads. As a result, in the worst case, approximately

every 5 out of 18 pads can be moved. With the total number of pads as $P_1 \times P_2$, the number of movable pads is:

$$N_M = \frac{5}{18} P_1 \times P_2 \tag{5.3}$$

Based on the above analysis, we can see that approximately $\frac{1}{3} \sim \frac{1}{2}$ number of pads can be moved each time.

5.1.3 Internal and external pads of low IR-drop area

5.1.3.1 Fast escaping power pads from low IR-drop areas to high IR-drop areas

As shown in Figure 5.6, there are a lot of excessive power pads in the low IR-drop area. With the above pad moving strategy, these pads will be propagated from the left-bottom area to other areas in a wave format. The outmost layer of pads of the left-bottom area moves toward other areas first. Then, the inner layer of pads starts to propagate when they become the new outmost layer of pads in this area. It may take several iterations before we can relocate the excessive power pads in the low IR-drop area to other areas.

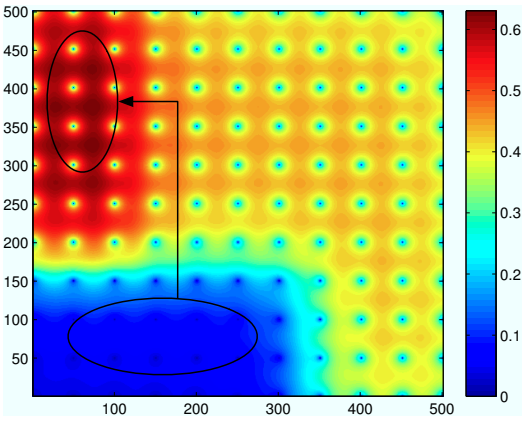


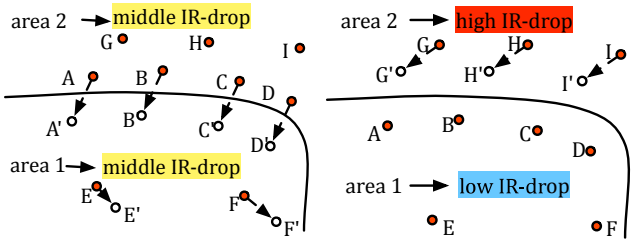
Figure 5.6: Fast escaping pads of low IR-drop areas

A simple but effective method to accelerate the above process is as follows. We extract the power pads located at low and high IR-drop values by comparing the average IR-drop values of their control areas. We gather the average IR-drop values for all the control areas of the power pads. We utilize the medium IR-drop value of all the averaged values as a reference.

Power pads whose average IR-drop value of its control area is less than 10% or higher than 90% of the reference value are assumed the pads in low and high IR-drop areas. With this information, we directly move the pads in low IR-drop areas to the neighboring candidate locations of the pads with high IR-drop areas, which is shown in Figure 5.6.

5.1.3.2 Preventing external pads from moving to low IR-drop areas

In the late stage of optimization, most pads have escaped from the low IR-drop area. Correspondingly, the low IR-drop area becomes the middle IR-drop area, such as the bottom area in Figure 5.7(a). Assume new locations of pads $A - D$ are located in the bottom area. After moving the pads, the bottom area becomes a low IR-drop area again, while the top area becomes a high IR-drop area.



(a) before (b) after

Figure 5.7: Before and after moving external pads of the low IR-drop area

We notice that the power grid is generally uniform in interconnection. The low IR-drop area is produced because currents extracted from that area are low. If the areas are in rectangle shape, with the provided current distribution, we can locate the bounding box of the low IR-drop area and forbid the external pads from moving to the area. With a more complex shape of the areas, the placement of the device blocks is required to define the bounding boxes of areas.

With the above mentioned techniques, the proposed pad placement optimization algorithm is listed in Algorithm 4. P is a fixed number of iteration.

Algorithm 4: Pad placement optimization algorithm

```
1: while iter < K(K > P) or Max_IR decreases within M1 iterations do
2:    Calculate new locations of all pads.
3:    // Move pads along constraint graph
4:    while At least one pad is not marked with M or F do
5:       if control area of pad A has highest IR-drop values then
6:          Mark A with M and neighboring pads of A with F.
7:          Move A to its new location.
8:       end if
9:    end while
10:   if iter >= P then
11:      Apply the pad escaping and blocking techniques.
12:   end if
13:   Keep track of best pad placement.
14:   Max_IR = min(Max_IR).
15: end while
```

5.2 Experimental results

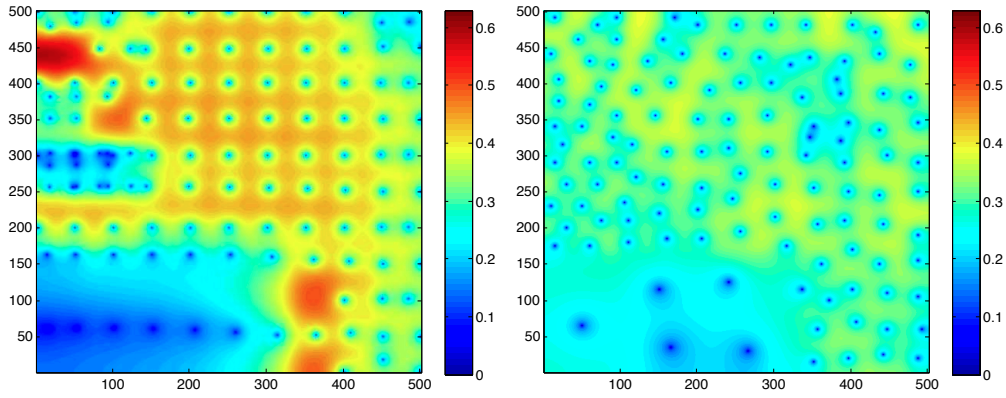
The proposed method is developed in C++ and tested on a set of artificially generated benchmarks with flip chip packages. The program is tested on an Intel CPU with a frequency of 3.33 GHz and 32 GB RAM.

5.2.1 IR-drop reduction effect of the proposed method

We first verify the IR-drop reduction effect of the proposed method with a test case with 250 K nodes. Initial IR-drop distribution of the test case is listed in Figure 5.1. The 121 pads are in uniform distribution. With the proposed method, pads in the low IR-drop area are moved to the high IR-drop area. With graph constraints, pads in other areas are moved to the calculated new locations.

Figure 5.8(a) shows the updated IR-drop distribution after the first iteration. With the technique of fast escaping pads from low IR-drop areas, most of the pads in the left-bottom area in Figure 5.8(a) are moved to other high IR-drop areas. To stop pads of other areas from moving to the left-bottom area, we keep the left-bottom area blocked for pads outside of this area. We keep track of the maximum IR-drop value of the grid for each iteration and expect it to keep decreasing in the optimization process. If it is not further

decreased within some iteration, we assume convergence is reached. Final pad placement is the one with a minimum of all the tracked maximum IR-drop values. The optimization process for the test case in Figure 5.1 costs 12 iterations. Optimized pad and IR-drop distributions are shown in Figure 5.8(b). Comparing the optimized IR-drop distributions and the initial case, we can see that there is a great reduction in the maximum IR-drop value. Standard deviation of the grid also gets better, which is reflected from the much more even distribution of the IR-drop values.



(a) 1st iteration (b) 12 iterations

Figure 5.8: Pad and IR-drop distributions of a power grid with 250 K nodes

5.2.2 Comparison of the proposed method vs. others

5.2.2.1 Modified SA method

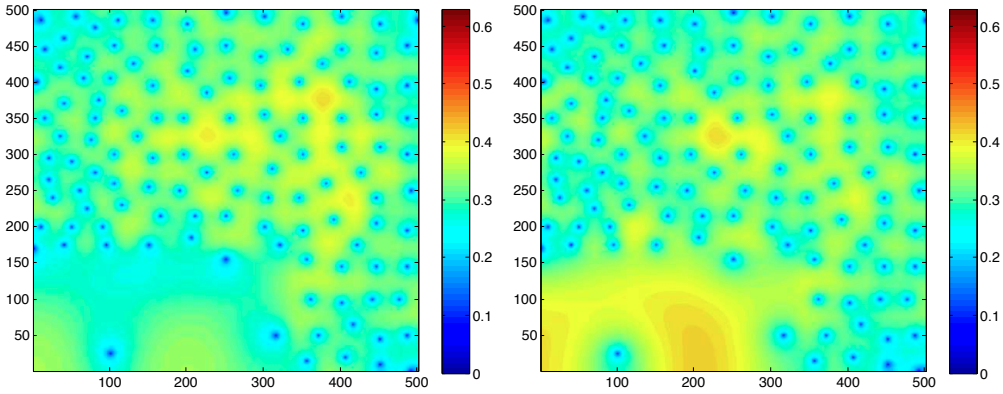
To fully evaluate the proposed method, we first develop a modified Simulated Annealing (SA) method [12].

Similar to the idea of the fast escaping technique, we first move the pad located in the lowest IR-drop area to the neighboring candidate of pads of the highest IR-drop areas. After each movement, we update IR-drop values of the whole grid with CHOLMOD [29]. If the maximum IR-drop value is less than the previous IR-drop value, we continue the moving process. If not, we stop the operation.

After the preprocessing step, we then perform the SA method. At each temperature, we try moving pads M times. In each of the moves, a pad

located in the low IR-drop areas is selected and relocated to one of its neighboring candidate locations of the power pads. After relocating the pad, voltage values of nodes around the old and new locations of the moved pad are updated with the Gauss-Seidel method. The local area covers nodes whose old and new voltage values are different more than ϵ. In our experiment, we set $\epsilon = 1^{-10}$.

The final IR-drop distribution with the modified SA method is shown in Figure 5.9(a). We verify the correctness of the final solution by implementing CHOLMOD with the final pad placement. The verified final IR-drop distribution is shown in Figure 5.9(b).



(a) final results with modified SA (b) corrected final results

Figure 5.9: Optimization results with modified SA method

Comparing Figure 5.9(a) and Figure 5.9(b), we can clearly see that there is an error in the modified SA method. Despite enlarging the updating area with very small ϵ, the accumulated error during the optimization process still cannot be ignored. Maximum IR-drop values in Figure 5.9(a) and Figure 5.9(b) are 0.397 V and 0.401 V respectively. With other test cases or longer simulation, the error may be larger. More seriously, because of this error, the decision for the pad selection and relocation during the optimization process may not be correct.

Besides error, the modified SA method is slow. This is because the local IR-drop values are updated with each movement of a single power pad. The modified SA method costs 7 minutes to get the results in Figure 5.9(a). Although the runtime for one updating can be less with bigger ϵ, the resulting error will be larger, which leads to more unreliable results.

5.2.2.2 MG_SA method

In order to eliminate the error of the modified SA method, we can perform static power grid analysis of the whole grid after each movement of a pad. Considering that this operation is very expensive, especially for a large size power grid, we accelerate the modified SA method with a multigrid technique. The new method is referred as MG_SA.

We restrict the power grid and candidate grid into coarser layers and start the pad optimization from the coarsest layer. Then, we project optimized locations of the pads into finer layers and perform another pad optimization process. This process goes on until the optimization process on the finest grid is done. After each movement of a pad, we perform CHOLMOD to get new IR-drop values. To protect the global behavior of the grid, we build coarse grids with a similar method as [4,9]. Due to the similarity of global behavior between coarse and fine grids, minimizing IR-drop values on a coarser grid through pad placement also helps reduce IR-drop values on a finer grid. Because the simulation cost on a coarse layer is much less, we perform more intense pad movements on these layers, so that we can reach the same results with less effort.

Table 5.1 shows the simulation results of MG_SA and the proposed method. Because there is error in the final results of the SA method, for fair comparison, we do not compare this with the SA method. The test cases are artificially generated 2D power grids, with 1.4 V supply voltage. The large IR-drop values may not represent real power grid designs, and they are just generated to illustrate the effectiveness of the proposed method. Data with the proposed method is obtained with 10-14 iterations. Three layers are utilized in the MG_SA method.

We can also see that with the two methods, maximum IR-drop and standard deviation are much smaller than the initial case. The proposed method outperforms the MG_SA method with much less runtime and the same level of IR-drop distribution. To give a more specific comparison of the different methods, we utilize $C250$ as an example. It is also the test case for the modified SA method. For $C250$, the modified SA method costs 7 minutes to reduce the maximum IR-drop to 0.4 V; MG_SA costs 5 minutes and 12 seconds to decrease the maximum IR-drop to 0.383 V; the proposed method costs only 31 seconds to decrease the maximum IR-drop value to 0.391 V.

Table 5.1: Simulation results of MG_SA vs. proposed method

ckt	Method	N_p	IR_b	IR_a	σ_b	σ_a	time
C250K	MG_SA	121	0.631V	0.383V	0.152	0.041	5m12s
	Graph			0.391V		0.046	31.7s
C640K	MG_SA	441	0.243V	0.158V	0.053	0.016	17m35s
	Graph			0.152V		0.014	1m23s
C1M	MG_SA	121	0.396V	0.261V	0.085	0.024	30m
	Graph			0.247V		0.024	2m47s
C1.6M	MG_SA	231	0.388V	0.280V	0.087	0.030	90m26s
	Graph			0.216V		0.020	6m55s

MG_SA: multigrid accelerated modified SA method; Graph: the proposed method; N_p: number of pads; IR_b and IR_a: maximum IR-drop before and after optimization; σ_b and σ_a: standard deviation of IR-drop before and after optimization; time: total runtime.

Table 5.2: Proposed method on larger test cases

ckt	N_p	iter	IR_b	IR_a	σ_b	σ_a	time
C2M	231	12	0.412V	0.256V	0.094	0.023	11m42s
C4M	441	14	0.415V	0.275V	0.138	0.024	35m12s
C8M	861	10	0.416V	0.292V	0.109	0.031	81m10s
C16M	441	12	0.188V	0.128V	0.047	0.011	118m50s

For other test cases in Table 5.1, the proposed method is able to reach better IR-drop values with much less runtime. The proposed method is very effective in performing the optimization process.

Table 5.2 shows the simulation results of the proposed method with some larger test cases. Considering the MG_SA is much slower than the proposed method, especially for these larger test cases, we only list the results of the proposed method. With only tens of iterations, the proposed method generates an optimized placement of pads, which leads to a much better IR-drop distribution. Runtime of the proposed method is also very affordable.

Most of the runtime is the cost of performing the power grid analysis in each iteration. It can be reduced with more advanced techniques such as multigrid [9], domain decomposition [30] and parallel method [14]. For example, it only costs 16 seconds for [14] to perform power grid analysis to a power grid with 16 M nodes, while CHOLMOD costs 7 minutes for the same operation. With the technique of [14], we can instantly reduce the runtime for $C16M$ from 113 minutes to 30 minutes.

5.2.3 Convergence analysis

Figure 5.10 shows the convergence curves of maximum IR-drop values and standard deviation of IR-drop values versus the iteration number. Test case $C250K$ is utilized for illustration. With a very limited number of iterations, the proposed optimization method reaches convergence. Standard deviation values exhibit the same trend with that of maximum IR-drop value.

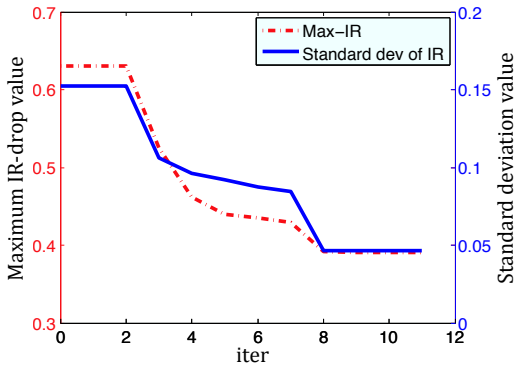


Figure 5.10: Convergence test of the optimization method

5.3 Summary

In this chapter, we propose an efficient method to optimize the locations of a fixed number of power pads. We develop a novel IR-drop driven method to calculate new locations of all the pads. Moving pads to these new locations reduces local IR-drop values. We develop a graph-based strategy to move power pads to reduce global IR-drop values. In each iteration, multiple power pads are relocated, which follows by a power grid DC analysis to update the IR-drop values of the grid. Experimental results show that the proposed method is very effective in reducing the IR-drop values.

CHAPTER 6

EFFICIENT SIMULATION-BASED OPTIMIZATION OF POWER GRID WITH ON-CHIP VOLTAGE REGULATOR

As stated in the introduction, the on-chip low-dropout voltage regulator (LDO) has a much smaller size than that of off-chip voltage regulators. The benefit of utilizing on-chip LDOs to connect the global and local power grids is to reduce both high-frequency on-chip power grid noises and mid-frequency noises caused by package variation [14]. It is promising to integrate the LDOs into the power grid and meet the 10% worst IR-drop constraint. The power grid model with the LDOs is shown in Figure 6.1.

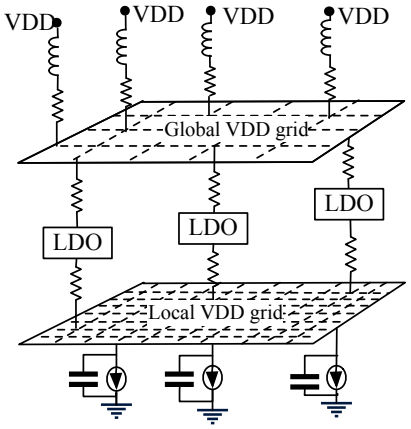


Figure 6.1: Power grid model with LDO integration

In Figure 6.1, the power grid is represented as an RLC network. LDOs connect the global and local power grids. In this chapter, we focus on decreasing the on-chip noises of the VDD network with flip chip technology. Only a single power domain is considered.

In order to address the challenges in reducing IR-drop values through inserting the on-chip LDOs, we first propose an efficient hybrid method to simulate a power grid with LDOs. The Cholesky direct solver and SPICE are utilized to get the voltage values of power grid nodes and LDOs' input / output nodes. Based on the simulation method, we propose an efficient flow

to add a set of LDOs into power grids to meet the IR-drop constraint. We divide the power grid into global and local power grids. In order to save the runtime, the local power grid is majorly utilized to optimize the LDOs. The LDOs are settled at the locations where the IR-drop values are effectively reduced. By testing the method on a set of benchmarks and comparing it with two other methods, we verify the effectiveness of the proposed method.

In the rest of this chapter, Section 6.1 to Section 6.3 present the proposed method. Experimental results are listed in Section 6.4. Future works and summary are listed in Section 6.5 and Section 6.6.

6.1 LDO's feature and supply voltage of power grid

Figure 6.2(a) shows the model of an external capacitor free low-dropout voltage regulator (LDO) [31]. It is composed with a pass element (M_p), sampling resistors (R_{f1}) and (R_{f2}), reference voltage (V_{ref}), an error amplifier and a differentiator. We adopt the LDO topology from [31] and modify the design to generate an output voltage of 1.8 V. With 100 mA output current, the $V_{in} - V_{out}$ curve of the LDO is shown in Figure 6.2(b). It is based on 250 nm technology. The 1.8 V output voltage may be too high for the current on-chip power supply system. With well-designed transistor model of the voltage regulator, it is possible to generate low output voltage, i.e. less than 1 V. However, the design of the transistor model requires a lot of considerations such as transient response and so on. We didn't find other public transistor model of the regulator that outputs low voltage. In this dissertation, we perform the exploration based on the voltage regulator model from [31]. The output voltage is 1.8 V.

As shown in Figure 6.2(b), LDO works in three different regions: off region, dropout region and regulation region. In dropout region, the output voltage of LDO is a linear function of input voltage. As the input voltage increases, LDO starts to work in the regulation region, where the output voltage is constant and is almost not affected by the variation of the input voltage.

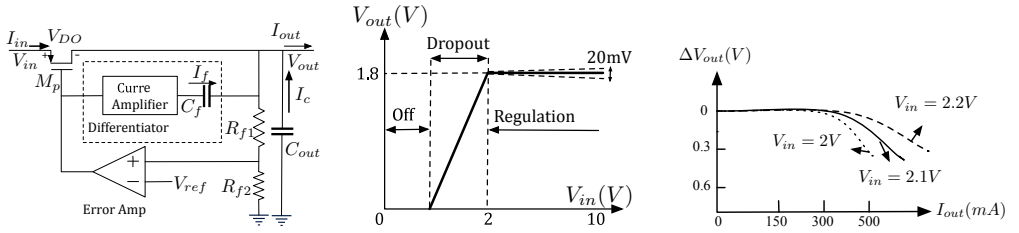
Due to the resistance of M_p, there is a voltage drop between the input and output voltage of LDO when it starts to work in the regulation region. For example, Figure 6.2(b) shows that when the LDO starts to work in the regulation region, $V_{in} = 2$ V and $V_{out} = 1.8$ V. The voltage drop between V_{in}

and V_{out} is 200 mV.

LDO's output voltage depends on its input voltage and output current. Utilizing V_{out} = 1.8 V as a reference, Figure 6.2(c) shows the changes of LDO's output voltages with different output currents and input voltages. As output current increases, output voltage starts to decrease. The larger the output current, the less the output voltage. Increasing input voltage helps to prevent output voltage from decreasing.

With the above observations, we should increase LDO's input voltage to make it less affected by big output currents. Input voltage cannot be too high, as higher input voltage reduces LDO's power efficiency [14]. Combining these considerations, we set V_{in} = 2.2 V.

Because the LDOs obtain voltage from the global grid, the power supply of the global power grid should be at least 2.2 V. In the rest of this chapter, we set the power supply of the global power grid as 2.2 V. Ideally, the local power grid obtains 1.8 V voltages from the LDOs. The voltage values of the local grid are directly affected by the distribution of the LDOs. In order to satisfy the IR-drop constraint, we need to optimize the number and locations of the LDOs.



(a) LDO structure

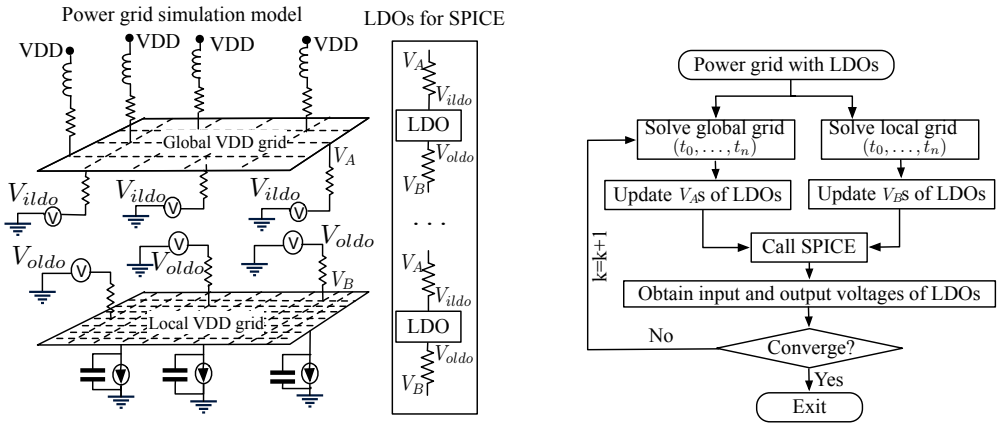
(b) $V_{in} - V_{out}$ curve of LDO

(c) $\Delta V_{out} - I_{out}$ curves with different V_{in}

Figure 6.2: Structure and electrical characteristic curves of an LDO

6.2 Hybrid simulation of power grid with LDOs

Because there is no public numerical model for LDO, a traditional power grid solver such as SuperLU and PCG cannot simulate the system of the power grid and LDOs. Based on the Cholesky direct solver and SPICE, we propose a method to perform the simulation.



(a) Simulation models of power grid and LDOs (b) Simulation flow

Figure 6.3: Hybrid simulation of power grid with LDO integration

The simulation models of the power grid and LDO are shown in Figure 6.3(a). In the power grid simulation model, LDOs are treated as voltage sources. The values of the voltage sources are the LDOs' input and output voltages. LDOs are simulated by SPICE with the model in Figure 6.3(a). In Figure 6.3(a), V_{ildo} and V_{oldo} are the input and output voltages of LDOs. V_A and V_B are the voltage values of nodes in global and local power grids.

The simulation flow is illustrated in Figure 6.3(b). Starting with V_{ildo} = 2.2 V and V_{oldo} = 1.8 V, we simulate the global and local power grids to obtain V_As and V_Bs. In the simulation, companion models of capacitors and inductors are adopted from [28]. "CHOLMOD" is utilized as the Cholesky solver. After this, SPICE is utilized to simulate the LDOs and update all the input and output voltages of the voltage regulators. This process repeats until the voltage values of power grid nodes are converged.

6.3 Optimizing power grid with LDOs

Based on the simulation method, we propose an effective flow to optimize the LDOs to meet the IR-drop constraint.

6.3.1 Optimization flow

The optimization flow is shown in Figure 6.4. With the geometrical information of device blocks, we extract candidate locations of the LDOs. For power grid, initially, we assume there is a single randomly settled LDO connecting the global and local power grids. Based on the DC analysis of local power grid, we develop a method to set the LDO at a location, where the IR-drop values can be effectively reduced. The optimization improves DC operation point of the circuit, which helps reduce IR-drop values of the transient time steps.

After optimizing the single LDO, we perform transient analysis of the local power grid. A set of LDOs is added to eliminate the IR-drop violations. We develop an efficient method to find the locations of the new LDOs. Adding LDOs at those locations reduces IR-drop values very effectively. Each newly added LDO is assumed to provide an ideal output voltage to the local power grid. Every LDO is connected to the global power grid through the closest node of the global power grid to the LDO. After adding the LDOs, we verify the power grid with LDOs with our simulation method. Accurate output voltages of LDOs are obtained at this step. New iteration starts if there is still an IR-drop violation.

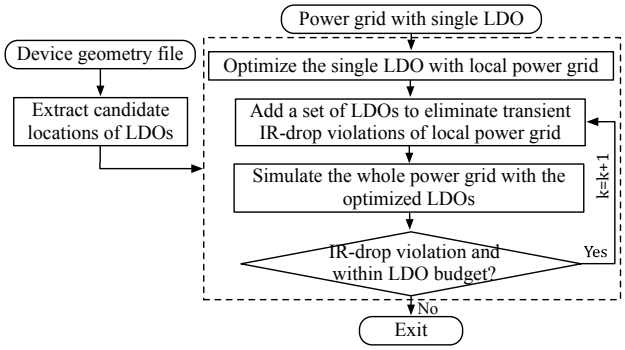


Figure 6.4: Power grid optimization flow with LDOs

In the flow, we only simulate the local power grid to optimize the LDOs. The insertion of the new LDOs depends on the voltage values of the power grid nodes, which are obtained through transient analysis. Because both the voltages of global and local power grid nodes are affected by the LDOs, we have to go through iterations of transient analysis of the whole power grid and SPICE simulations to obtain accurate voltage values of the power grid

nodes and LDOs. In our flow, we alleviate the iterations of transient analysis and SPICE simulation by approximating the voltage values of the LDOs. LDOs that are new to the optimization loop have ideal output voltages. Other LDOs utilize their output voltages from previous iterations of the optimization loop. A set of LDOs is inserted by only performing transient analysis of the local power grid, which makes the optimization much faster. Besides, by assuming that each new LDO provides an ideal output voltage to the local power grid, we are conservative about the optimization results. Because real output voltages of the LDOs are always lower than the ideal ones, the risk of adding excessive LDOs is reduced.

6.3.2 Candidate locations of LDOs

Assuming a LDO as a rectangle, we divide the chip into grids by the unit of the LDO's width and height. The grids are shown as the dotted small rectangles in Figure 6.5. Grids that are not occupied by device blocks are the candidate locations of LDOs.

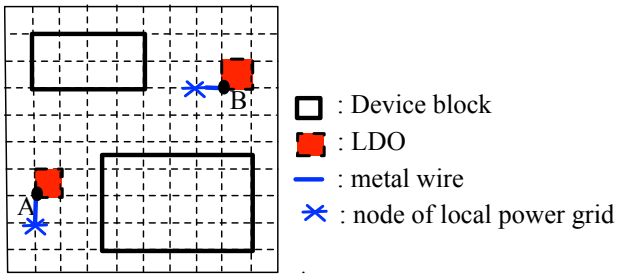


Figure 6.5: Candidate locations of LDOs

We represent a grid with its bottom-left corner node. For example, in Figure 6.5, node A and B represent the rectangles which are occupied by the LDOs. The star symbols around the LDOs represent the nodes of the local power grid. The lines connecting the LDOs and the nodes are the metal wires. LDOs should be close to the nodes to reduce the wire lengths.

6.3.3 Optimizing the location of the single LDO

The process of optimizing the single LDO is shown in Figure 6.6. We develop an effective method to calculate the new location of the LDO. DC analysis of

the local power grid is performed after moving the LDO to the new location. After the DC analysis, we record the maximum IR-drop value of the local power grid and the new location of the LDO. After repeating this process for M iterations, we recover the LDO to the location that results in the smallest maximum IR-drop value. In our experiment, $M = 5$.

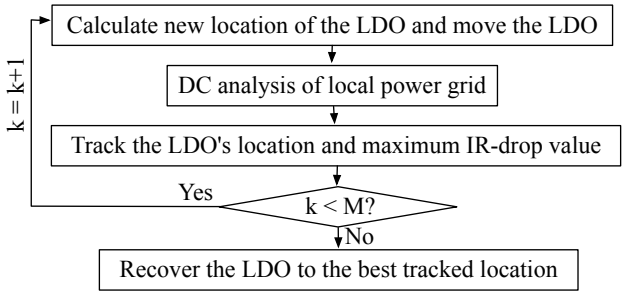


Figure 6.6: Optimization flow of the single LDO

The new location of the LDO is calculated as follows. With DC IR-drop values of local power grid nodes, similar to [32], we assign each node a weight, which is proportional to its IR-drop value. We then calculate the weighted geometrical center of local power grid nodes. The LDO should be moved to the closest candidate location to the geometrical center. In order to shorten the wire length from the LDO to the local power grid, the LDO should connect to the local grid node that is closest to the geometrical center.

The above method effectively counts in the severity of different IR-drop values, which are weighted by their values and the number of nodes with the IR-drop values. Setting the LDO at the new location not only reduces the maximum IR-drop value, but also results in a more even IR-drop distribution. By moving the LDO a few times, the single LDO can be quickly settled at a location where IR-drop values of the local power grid are effectively reduced.

6.3.4 Adding LDOs to eliminate IR-drop violations

In previous step, we optimize the location of the single LDO. Depending on the devices' switching activities and current extractions, more LDOs may be required to meet the IR-drop constraint.

Our flow of adding LDOs is shown in Figure 6.7. First, we perform transient analysis of the local power grid. If there is an IR-drop violation, one

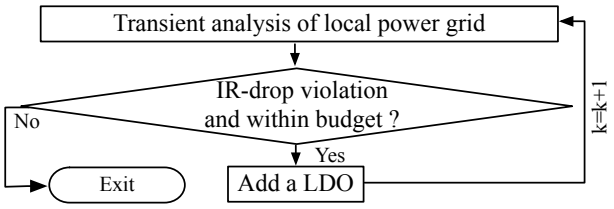


Figure 6.7: Flow of adding LDOs to eliminate IR-drop violations

LDO is added. This process repeats until the IR-drop constraint is met or there is no space to insert another LDO.

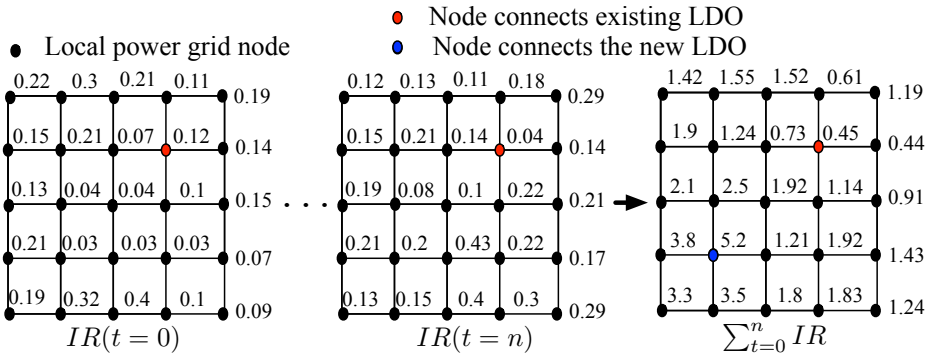


Figure 6.8: Accumulating IR-drop values for local power grid nodes

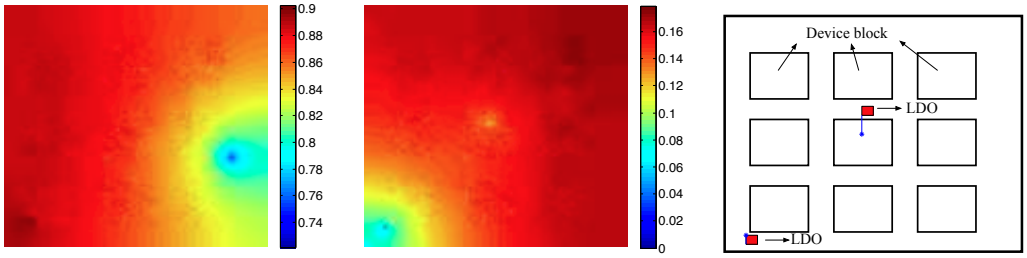
We develop a very effective method to add the new LDO. As shown in Figure 6.8, while performing transient analysis, we record the IR-drop values of the nodes of the local power grid. By the end of the analysis, we obtain the accumulated IR-drop value of each node. The new LDO is settled at the closest candidate location to the node with maximum accumulated IR-drop value, such as the blue node of Figure 6.8.

Setting the LDO based on the accumulated IR-drop values counts in the severity of both the devices' switching activities and current extractions along all the time steps. Each added LDO is able to effectively reduce the IR-drop values of the local power grid.

The pseudo code of the whole optimization process is listed in Algorithm 5. V_{oldo_ideal} is the ideal output voltage value of LDO. $V_{oldo\_ideal} = 1.8$ V.

Algorithm 5: Optimize power grid with LDOs

```
1  begin
2      Optimize the initial LDO with DC local power grid analysis;
3      while (1) do
4          while (1) do
5              Perform transient analysis of local power grid;
6              ΔV_max ← worst-case maximum IR-drop value;
7              if (ΔV_max ≤ 10% × V_oldo_ideal or no space for new LDO) then
8                  break;
9              end
10             Add a LDO;
11         end
12         Solve the whole power grid and the LDOs with the simulation method;
13         if (ΔV_max ≤ 10% × V_oldo_ideal or no space for new LDO) then
14             break;
15         end
16     end
17 end
```



(a) Worst-case IR-drop values before optimization

(b) Worst-case IR-drop values after optimization

(c) Locations of optimized LDOs

Figure 6.9: Worst-case IR-drop values and locations of LDOs with the proposed optimization method

6.4 Experimental results

The program is developed with C++ and SPICE. It is executed on a single CPU with a frequency of 2.13 GHz, 512 KB cache and 32 GB RAM. We generate several test cases to test the optimization flow. Each test case includes a power grid file and a device geometry file. The device geometry file specifies the shape and locations of device blocks and the initial location of the single LDO.

The test cases have similar parameters as that of IBM power grid benchmarks [24]. There are 1000 time steps in the transient analysis. We generate pulse current sources to represent devices' switching. The period of the currents varies from 1 ns to 10 ns. We also assign small decoupling capacitors

with the devices, whose values vary from 0.01 μF to 0.1 μF. These capacitors are to compensate for the fast switching of devices.

In our experiment, the convergence of the proposed simulation method happens when the worst-case voltage differences of all the nodes between two continuous iterations are less than $3e^{-4}$ V. Because LDOs provide the local power grid power with an ideal output voltage of 1.8 V, the IR-drop constraint is:

$$\Delta V_{max} \leq 0.1 \times 1.8V = 0.18V \tag{6.1}$$

6.4.1 Effect of the proposed optimization method

We test the effect of the optimization flow on a small size power grid. There are around 17 K nodes in the local power grid. The global power grid has a similar number of nodes. The initial LDO is located at the right-hand side of the chip. The maximum DC IR-drop value is 0.206 V. With the initial LDO, worst-case IR-drop values are shown in Figure 6.9(a). The maximum IR-drop value is 0.913 V, which seriously violates the IR-drop constraint.

In the optimization, the LDO is moved to the center of the chip, which decreases the DC maximum IR-drop value to 0.158 V. The IR-drop constraint is met by adding only one more LDO. The locations of two LDOs are shown in Figure 6.9(c). With the two LDOs, worst-case IR-drop values are shown in Figure 6.9(b). The maximum IR-drop value is 0.179 V.

We randomly pick up a node of the local power grid and record its voltage values before and after the optimization, which are shown as the red and blue curves in Figure 6.10, respectively. The nodes' voltage values are greatly increased by performing the optimization.

By comparing the IR-drop values and nodes' voltage values before and after the optimization, we can see that our method is very efficient in meeting the IR-drop constraint.

6.4.2 Comparison of the proposed method and others

Utilizing the above test case, we compare our proposed method with two other methods. The first one assumes an even distribution of LDOs. The

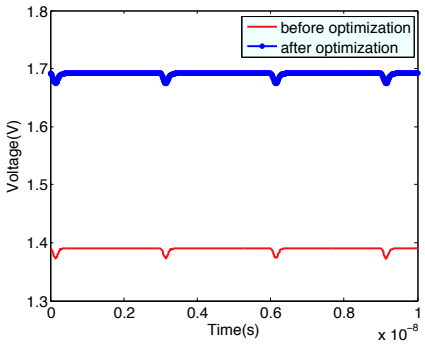


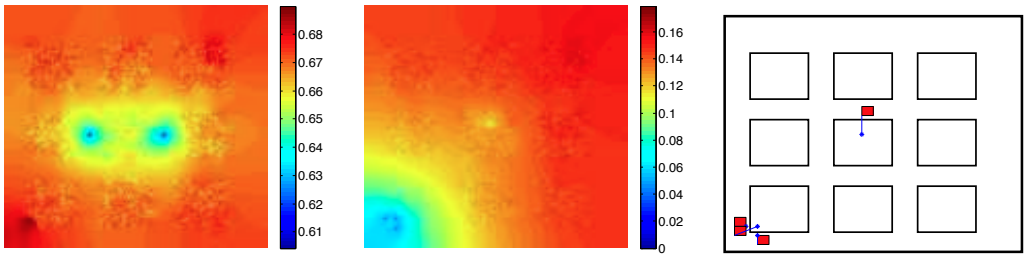
Figure 6.10: Voltage values of a node before and after optimization

second method adds LDOs with a different strategy.

We first compare the proposed method with an even distribution of LDOs. Proven by our proposed method, two LDOs are sufficient to meet the IR-drop constraint. For fairness, we generate two LDOs and evenly distribute them in the whitespace. The maximum IR-drop value is 0.692 V, which is much higher than our method. Worst-case IR-drop values are shown in Figure 6.11(a). Comparing the IR-drop values to our optimization results, which are shown in Figure 6.9(b), we can see that our method achieves much smaller IR-drop values by allowing uneven distribution of LDOs.

In the second comparison, we develop another method to add the LDOs. Instead of adding a LDO after the complete transient analysis of the local power grid, the LDO is added when there is an IR-drop violation at any time step. The new LDO is settled at the closest candidate location to the local power grid node, which has the maximum IR-drop value of the time step. With the method, three LDOs are added, which are shown in Figure 6.11(c). Worst-case IR-drop values are shown in Figure 6.11(b). The optimized maximum IR-drop value is 0.163 V.

Comparing Figure 6.11(b) and Figure 6.9(b), we see that the second method requires more LDOs than ours to meet the IR-drop constraint. The reason is as follows. In the second method, a LDO is added with the appearance of IR-drop violation at any time step. Adding the LDO improves the DC operation point of the transient analysis, which results in higher real voltages than the ones obtained by continuing simulating the later time steps. The method does not count in this factor and continues adding LDOs based on the worse IR-drop values than real ones, which results in excessive LDOs to meet the IR-drop constraint.



(a) Worst-case IR-drop values with even distribution of LDOs

(b) Worst-case IR-drop values of the second comparison method

(c) Optimized LDOs of the second comparison method

Figure 6.11: Worst-case IR-drop values and locations of LDOs of the comparison methods

6.4.3 Simulation results of more test cases

We test our optimization flow on several bigger benchmarks. A summary of the simulation results is listed in Table 6.1. The names of the test cases indicate the number of nodes in the local power grid. The global grid has a similar number of nodes. Meanings of other items are listed under Table 6.1.

Table 6.1: Simulation results of the proposed optimization flow

Case	N_{ldo}	N_B	ΔV_{imax}	ΔV_{omax}	$T(h:m:s)$	K
C17K	2	301	0.913V	0.179V	00:49:20	98
C525K	3	580	0.935V	0.155V	08:54:50	53
C1M	13	139	1.534V	0.167V	34:38:24	92
C2M	21	621	1.192V	0.173V	62:30:12	103

1. N_{ldo}: number of LDOs; 2. N_B: number of LDO candidates; 3. ΔV_{imax} and ΔV_{omax}: maximum IR-drop value before and after optimization; 4. $T(h:m:s)$: runtime (hour:minute:second); 5. K: total number of transient analysis of global or local power grid

With the initial LDO, maximum IR-drop values of the test cases vary from 0.913 V to 1.534 V, which greatly violate the IR-drop constraint. The violations are effectively removed by adding a set of LDOs with the optimization method. Optimized IR-drop values all meet the IR-drop constraint.

Table 6.1 also shows that the runtime increases a lot with bigger test cases. Over 90% of runtime is cost on the local and global power grid transient analysis, which are utilized to insert the LDOs and to obtain the voltage values of power grid nodes and LDOs. Considering there are 1000 time steps,

performing the multiple times of transient analysis results in long runtime, especially for bigger test cases. Runtime can be reduced by accelerating the transient analysis.

6.5 Future works

One of common methods to reduce power grid noises is through inserting decoupling capacitance. It would be meaningful to compare the effect of noise elimination by inserting decoupling capacitance or by adding on-chip low-dropout voltage regulator. However, it requires the re-formulation of the problem and developing a new algorithm to make fair comparison of the two methods. It is beyond the content of this dissertation but is a future work for those who are seeking for further understanding of power grid optimization.

6.6 Summary

In this chapter, we propose an efficient method to meet the IR-drop constraint of the power grid by optimizing the number and locations of on-chip low-dropout voltage regulators (LDO). The optimization is based on a hybrid simulation method, which is composed of the Cholesky solver and SPICE. Experimental results verify the effectiveness of the proposed method. To the best of our knowledge, this is the first work optimizing the number and locations of LDOs to meet the IR-drop constraint.

CHAPTER 7

CONCLUSIONS

We propose several works of power grid simulation and optimization to quickly obtain and reduce IR-drop values of the power grid.

We first propose several effective parallel methods to reduce the runtime of power grid DC and transient analysis. The methods are also capable of solving large size power grids. Based on the distributed memory system, we develop an efficient method for power grid DC analysis. With an efficient flow and data communication method, over 100X speedup is achieved for the power grid DC analysis. The power grid with 192 M nodes can be solved within a few minutes. Based on the shared memory system, we develop PGT_SOLVER to reduce the runtime of transient analysis. Advanced techniques are proposed and utilized in PGT_SOLVER, such as the sparse vector method and solution mapping technique. PGT_SOLVER is the fastest of the top three methods in the "TAU2012 power grid simulation contest". The runtime of transient analysis is effectively reduced. Combining the parallel DC method and the advanced techniques in PGT_SOLVER, we propose an efficient parallel solver for power grid transient analysis. Special considerations are made to reach better performance, such as power grid partitioning and so on. Experimental results show that the proposed method achieves over 69X speedup compared to sequential PGT_SOLVER. With the method, a power grid containing 32 M nodes can be processed within a reasonable runtime.

We then explore several methods to optimize the IR-drop values of the power grid, including the optimization of power pad locations and inserting low-dropout on-chip voltage regulators (LDO). We propose an efficient method to optimize the locations of a fixed number of power pads to reduce DC IR-drop values. Based on DC IR-drop analysis, we propose an effective method to calculate the locations of the pads. Moving the pads to these locations effectively reduces the maximum IR-drop value and results in a

more even IR-drop distribution. By building a pad connection graph, multiple power pads are moved to their calculated new locations. With a limited number of iterations, the optimization method effectively reduces the DC IR-drop values. We also explore the optimization of on-chip low-dropout voltage regulators to reduce transient IR-drop values. We propose a hybrid simulation flow to simulate the system of power grids with LDO integration. The Cholesky direct solver and SPICE are utilized to perform the hybrid simulation. We propose an effective optimization flow to add a set of LDOs at the locations, which effectively reduce transient IR-drop values. Experimental results verify the effectiveness of the method.

REFERENCES

[1] H. Qian, S. Nassif, and S. Sapatnekar, "Power grid analysis using random walks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 8, pp. 1204–1224, August 2005.

[2] T.-H. Chen and C. C.-P. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," in *Proceedings in Design Automation Conference*, 2001, pp. 559–562.

[3] C.-J. Wei, H. Chen, and S.-J. Chen, "Design and implementation of block-based partitioning for parallel flip-chip power-grid analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 3, pp. 370–379, March 2012.

[4] Z. Feng and P. Li, "Multigrid on GPU: Tackling power grid analysis on parallel SIMT platforms," in *IEEE/ACM International Conference on Computer-Aided Design, 2008*, Nov. 2008, pp. 647–654.

[5] J. W. Demmel, J. R. Gilbert, and X. S. Li, "An asynchronous parallel supernodal algorithm for sparse Gaussian elimination," *SIAM J. Matrix Analysis and Applications*, vol. 20, no. 4, pp. 915–952, 1999.

[6] V. Voronov and N. Popova, "Parallel power grid simulation on platforms with multi core processors," in *International Conference on Computing, Engineering and Information*, April 2009, pp. 144–148.

[7] PETSC, "The petsc website," http://www.mcs.anl.gov/petsc.

[8] K. Sun, Q. Zhou, K. Mohanram, and D. Sorensen, "Parallel domain decomposition for simulation of large-scale power grids," in *IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2007, pp. 54–59.

[9] J. Kozhaya, S. Nassif, and F. Najm, "Multigrid-like technique for power grid analysis," in *IEEE/ACM International Conference on Computer-Aided Design*, 2001, pp. 480–487.

[10] X. S. Li, J. Demmel, and J. Gilbert, "The SuperLU website," http://crd.lbl.gov/~xiaoye/SuperLU.

[11] C.-H. Lu, H.-M. Chen, and C.-N. J. Liu, "An effective decap insertion method considering power supply noise during floorplanning," *J. Inf. Sci. Eng.*, pp. 115–127, 2008.

[12] Y. Zhong and M. Wong, "Fast placement optimization of power supply pads," in *Proceedings of Asia and South Pacific Design Automation Conference*, January 2007, pp. 763–767.

[13] K. N. Leung and P. Mok, "A capacitor-free CMOS low-dropout regulator with damping-factor-control frequency compensation," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 10, pp. 1691–1702, October 2003.

[14] Z. Zeng, X. Ye, Z. Feng, and P. Li, "Tradeoff analysis and optimization of power delivery networks with on-chip voltage regulation," in *47th ACM/IEEE Design Automation Conference*, June 2010, pp. 831–836.

[15] S. Lai, B. Yan, and P. Li, "Stability assurance and design optimization of large power delivery networks with multiple on-chip voltage regulators," in *International Conference on Computer-Aided Design*, November 2012, pp. 247–253.

[16] A. Klawonn and L. F. Pavarino, "A comparison of overlapping Schwarz methods and block preconditioners for saddle point problems." *Numerical Linear Algebra with Applications*, vol. 7, pp. 1–25, 2000.

[17] T. Peng, K. Sertel, and J. L. Volakisu, "Convergence of a fully overlapping domain decomposition method," in *URSI General Assembly and Scientific Symposium of Internaltional Union of Radio Science*, August 2011.

[18] A. Grama, A. Gupta, and V. Kumar, "Isoefficiency: Measuring the scalability of parallel algorithms and architectures," *Parallel Distributed Technology: Systems Applications, IEEE*, vol. 1, no. 3, pp. 12–21, August 1993.

[19] V. Kumar and A. Gupta, "Analyzing scalability of parallel algorithms and architectures," *Journal of Parallel and Distributed Computing*, vol. 22, pp. 379–391, September 1994.

[20] Y. Zhong and M. Wong, "Fast block-iterative domain decomposition algorithm for IR drop analysis in large power grid," in *11th International Symposium on Quality Electronic Design*, March 2010, pp. 277–283.

[21] L. Grigori, J. W. Demmel, and X. S. Li, "Parallel symbolic factorization for sparse LU factorization with static pivoting," in *Second International Workshop on Combinatorial Scientific Computing*, 2005, pp. 1289–1314.

[22] Z. Li, R. Balasubramanian, F. Liu and S. Nassif, "2012 TAU Power Grid Simulation Contest: Benchmark Suite and Result," in *IEEE/ACM International Conference on Computer-Aided Design*, 2012.

[23] W. Tinney, V. Brandwajn, and S. Chan, "Sparse vector methods," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-104, no. 2, pp. 295–301, February 1985.

[24] "IBM Power Grid Benchmarks," http://dropzone.tamu.edu/~pli/PGBench.

[25] J. Yang, Z. Li, Y. Cai, and Q. Zhou, "Powerrush: Efficient transient simulation for power grid analysis," in *IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 653–659.

[26] X. Xiong and J. Wang, "Parallel forward and back substitution for efficient power grid simulation," in *IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 660–663.

[27] T. Yu, Z. Xiao, and M. D. F. Wong, "Efficient parallel power grid analysis via additive Schwarz method." in *IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 399–406.

[28] T. Yu and M. Wong, "PGT_SOLVER: An efficient solver for power grid transient analysis," in *IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 647–652.

[29] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 22:1–22:14, October 2008.

[30] Q. Zhou, K. Sun, K. Mohanram, and D. Sorensen, "Large power grid analysis using domain decomposition," in *Proceedings in Design, Automation and Test in Europe*, vol. 1, March 2006, pp. 1–6.

[31] R. Milliken, M. Silva, and S. Sanchez, "Full on-chip CMOS low-dropout voltage regulator," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 9, pp. 1879 –1890, September 2007.

[32] T. Yu and M. Wong, "A novel and efficient method for power pad placement optimization," in *International Symposium on Quality Electronic Design*, March 2013, pp. 158–163.